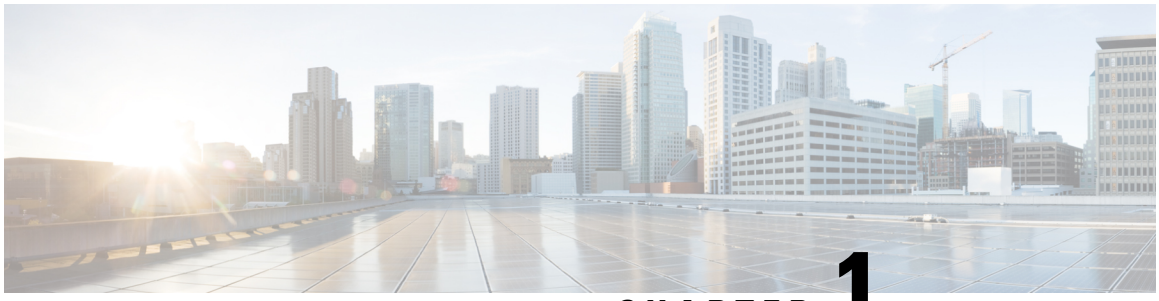
Routing Configuration Guide, Cisco IOS XE Gibraltar 16.10.x (Catalyst 9400 Switches)

First Published: 2018-12-10

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Configuring MSDP

Information About Configuring MSDP

This section describes how to configure the Multicast Source Discovery Protocol (MSDP on the switch. The MSDP connects multiple Protocol-Independent Multicast sparse-mode (PIM-SM) domains.

MSDP is not fully supported in this software release because of a lack of support for Multicast Border Gateway Protocol (MBGP), which works closely with MSDP. However, it is possible to create default peers that MSDP can operate with if MBGP is not running.

MSDP Overview

MSDP allows multicast sources for a group to be known to all rendezvous points (RPs) in different domains. Each PIM-SM domain uses its own RPs and does not depend on RPs in other domains. An RP runs MSDP over the Transmission Control Protocol (TCP) to discover multicast sources in other domains.

An RP in a PIM-SM domain has an MSDP peering relationship with MSDP-enabled devices in another domain. The peering relationship occurs over a TCP connection, primarily exchanging a list of sources sending to multicast groups. The TCP connections between RPs are achieved by the underlying routing system. The receiving RP uses the source lists to establish a source path.

The purpose of this topology is to have domains discover multicast sources in other domains. If the multicast sources are of interest to a domain that has receivers, multicast data is delivered over the normal, source-tree building mechanism in PIM-SM. MSDP is also used to announce sources sending to a group. These announcements must originate at the domain's RP.

MSDP depends heavily on the Border Gateway Protocol (BGP) or MBGP for interdomain operation. We recommend that you run MSDP in RPs in your domain that are RPs for sources sending to global groups to be announced to the Internet.

MSDP Operation

When a source sends its first multicast packet, the first-hop router (*designated router* or RP) directly connected to the source sends a PIM register message to the RP. The RP uses the register message to register the active source and to forward the multicast packet down the shared tree in the local domain. With MSDP configured, the RP also forwards a source-active (SA) message to all MSDP peers. The SA message identifies the source, the group the source is sending to, and the address of the RP or the originator ID (the IP address of the interface used as the RP address), if configured.

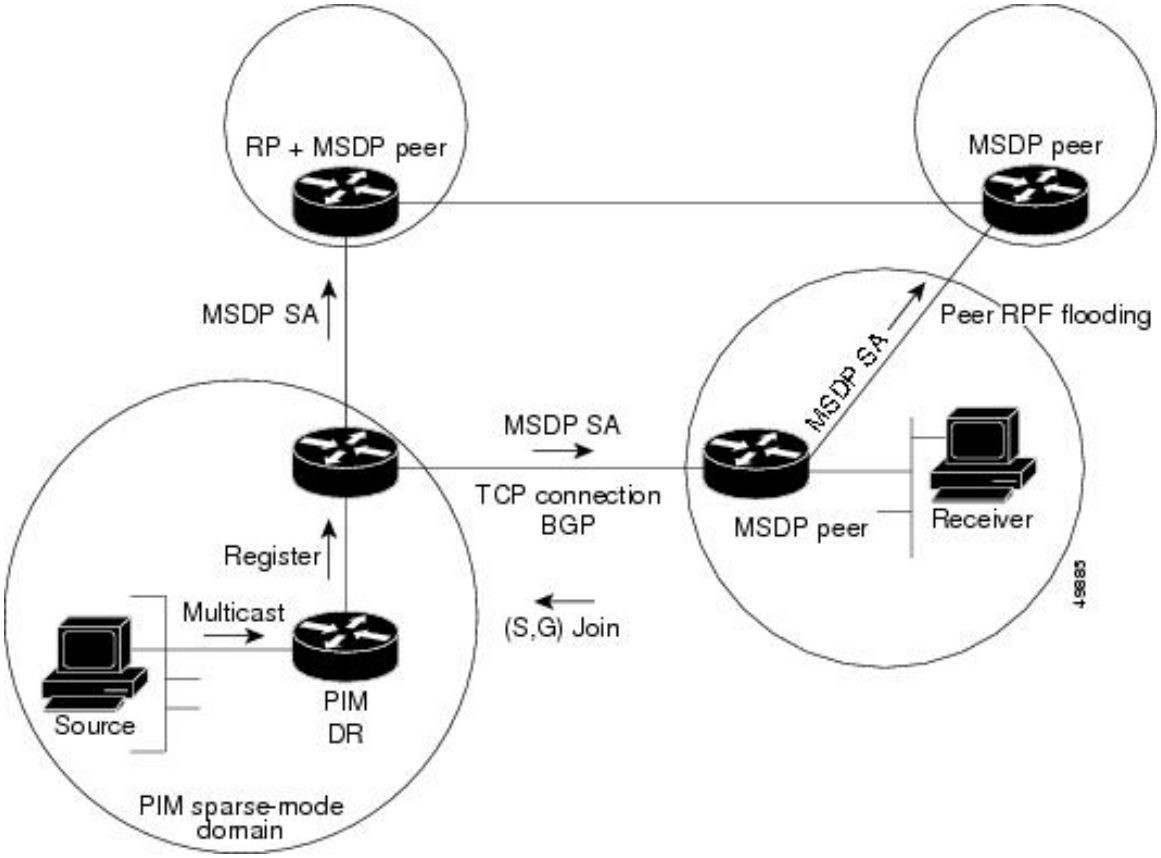
Each MSDP peer receives and forwards the SA message away from the originating RP to achieve peer reverse-path flooding (RPF). The MSDP device examines the BGP or MBGP routing table to discover which peer is the next hop toward the originating RP of the SA message. Such a peer is called an *RPF peer* (reverse-path forwarding peer). The MSDP device forwards the message to all MSDP peers other than the RPF peer. For information on how to configure an MSDP peer when BGP and MBGP are not supported, see the Configuring a Default MSDP Peer, on page 4.

If the MSDP peer receives the same SA message from a non-RPF peer toward the originating RP, it drops the message. Otherwise, it forwards the message to all its MSDP peers.

The RP for a domain receives the SA message from an MSDP peer. If the RP has any join requests for the group the SA message describes and if the (*,G) entry exists with a nonempty outgoing interface list, the domain is interested in the group, and the RP triggers an (S,G) join toward the source. After the (S,G) join reaches the source's DR, a branch of the source tree has been built from the source to the RP in the remote domain. Multicast traffic can now flow from the source across the source tree to the RP and then down the shared tree in the remote domain to the receiver.

Figure 1: MSDP Running Between RP Peers

This figure shows MSDP operating between two MSDP peers. PIM uses MSDP as the standard mechanism to register a source with the RP of a domain. When MSDP is configured, this sequence occurs.



By default, the switch does not cache source or group pairs from received SA messages. When the switch forwards the MSDP SA information, it does not store it in memory. Therefore, if a member joins a group soon after an SA message is received by the local RP, that member needs to wait until the next SA message to hear about the source. This delay is known as join latency.

Local RPs can send SA requests and get immediate responses for all active sources for a given group. By default, the switch does not send any SA request messages to its MSDP peers when a new member joins a group and wants to receive multicast traffic. The new member waits to receive the next periodic SA message.

If you want a new member of a group to learn the active multicast sources in a connected PIM sparse-mode domain that are sending to a group, configure the switch to send SA request messages to the specified MSDP peer when a new member joins a group.

MSDP Benefits

MSDP has these benefits:

- It breaks up the shared multicast distribution tree. You can make the shared tree local to your domain. Your local members join the local tree, and join messages for the shared tree never need to leave your domain.

- PIM sparse-mode domains can rely only on their own RPs, decreasing reliance on RPs in another domain. This increases security because you can prevent your sources from being known outside your domain.
- Domains with only receivers can receive data without globally advertising group membership.
- Global source multicast routing table state is not required, saving memory.

How to Configure MSDP

Default MSDP Configuration

MSDP is not enabled, and no default MSDP peer exists.

Configuring a Default MSDP Peer

Before you begin

Configure an MSDP peer.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp default-peer** *ip-address* | *name* [**prefix-list** *list*] **Example:** `Router(config)# ip msdp default-peer 10.1.1.1 prefix-list site-a`	Defines a default peer from which to accept all MSDP SA messages. • For *ip-address* | *name*, enter the IP address or Domain Name System (DNS) server name of the MSDP default peer. • (Optional) For **prefix-list** *list*, enter the list name that specifies the peer to be the default peer only for the listed prefixes. You can have multiple active default peers when you have a prefix list associated with each. When you enter multiple **ip msdp default-peer** commands with the **prefix-list** keyword, you use all the default peers at the same time for different RP

	Command or Action	Purpose
		prefixes. This syntax is typically used in a service provider cloud that connects stub site clouds. When you enter multiple **ip msdp default-peer** commands without the **prefix-list** keyword, a single active peer accepts all SA messages. If that peer fails, the next configured default peer accepts all SA messages. This syntax is typically used at a stub site.
Step 4	**ip prefix-list** *name* [**description** *string*] | **seq** *number* {**permit** | **deny**} *network length* **Example:** `Router(config)# prefix-list site-a seq 3 permit 12 network length 128`	(Optional) Creates a prefix list using the name specified in Step 2. • (Optional) For **description** *string*, enter a description of up to 80 characters to describe this prefix list. • For **seq** *number*, enter the sequence number of the entry. The range is 1 to 4294967294. • The **deny** keyword denies access to matching conditions. • The **permit** keyword permits access to matching conditions. • For *network length*, specify the network number and length (in bits) of the network mask that is permitted or denied.
Step 5	**ip msdp description** {*peer-name* | *peer-address*} *text* **Example:** `Router(config)# ip msdp description peer-name site-b`	(Optional) Configures a description for the specified peer to make it easier to identify in a configuration or in **show** command output. By default, no description is associated with an MSDP peer.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Caching Source-Active State

If you want to sacrifice some memory in exchange for reducing the latency of the source information, you can configure the Device to cache SA messages. Perform the following steps to enable the caching of source/group pairs:

Follow these steps to enable the caching of source/group pairs:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp cache-sa-state** [**list** *access-list-number*] **Example:** `Device(config)# ip msdp cache-sa-state 100`	Enables the caching of source/group pairs (create an SA state). Those pairs that pass the access list are cached. For **list** *access-list-number*, the range is 100 to 199. **Note** An alternative to this command is the **ip msdp sa-reques** global configuration command, which causes the Device to send an SA request message to the MSDP peer when a new member for a group becomes active.
Step 4	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* **Example:** `Device(config)# access-list 100 permit ip 171.69.0.0 0.0.255.255 224.2.0.0 0.0.255.255`	Creates an IP extended access list, repeating the command as many times as necessary. • For *access-list-number*, the range is 100 to 199. Enter the same number created in Step 2. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *protocol*, enter **ip** as the protocol name.

	Command or Action	Purpose
		• For *source*, enter the number of the network or host from which the packet is being sent. • For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. • For *destination*, enter the number of the network or host to which the packet is being sent. • For *destination-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Requesting Source Information from an MSDP Peer

If you want a new member of a group to learn the active multicast sources in a connected PIM sparse-mode domain that are sending to a group, perform this task for the Device to send SA request messages to the specified MSDP peer when a new member joins a group. The peer replies with the information in its SA cache. If the peer does not have a cache configured, this command has no result. Configuring this feature reduces join latency but sacrifices memory.

Follow these steps to configure the Device to send SA request messages to the MSDP peer when a new member joins a group and wants to receive multicast traffic:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp sa-request** {*ip-address* | *name*} **Example:** `Device(config)# ip msdp sa-request 171.69.1.1`	Configure the Device to send SA request messages to the specified MSDP peer. For *ip-address* | *name*, enter the IP address or name of the MSDP peer from which the local Device requests SA messages when a new member for a group becomes active. Repeat the command for each MSDP peer that you want to supply with SA messages.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Source Information that Your Switch Originates

You can control the multicast source information that originates with your Device:

- Sources you advertise (based on your sources)
- Receivers of source information (based on knowing the requestor)

For more information, see the Redistributing Sources, on page 9 and the Filtering Source-Active Request Messages, on page 11.

Redistributing Sources

SA messages originate on RPs to which sources have registered. By default, any source that registers with an RP is advertised. The *A flag* is set in the RP when a source is registered, which means the source is advertised in an SA unless it is filtered.

Follow these steps to further restrict which registered sources are advertised:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp redistribute** [**list** *access-list-name*] [**asn** *aspath-access-list-number*] [**route-map** *map*] **Example:** `Device(config)# ip msdp redistribute list 21`	Configures which (S,G) entries from the multicast routing table are advertised in SA messages. By default, only sources within the local domain are advertised. • (Optional) **list** *access-list-name*— Enters the name or number of an IP standard or extended access list. The range is 1 to 99 for standard access lists and 100 to 199 for extended lists. The access list controls which local sources are advertised and to which groups they send. • (Optional) **asn** *aspath-access-list-number*—Enters the IP standard or extended access list number in the range 1 to 199. This access list number must also be configured in the **ip as-path access-list** command. • (Optional) **route-map** *map*—Enters the IP standard or extended access list number in the range 1 to 199. This access list number must also be configured in the **ip as-path access-list** command. The Device advertises (S,G) pairs according to the access list or autonomous system path access list.

	Command or Action	Purpose

Step 4	Use one of the following:
	• `access-list`*access-list-number* {`deny`|`permit`} *source* [*source-wildcard*]
	• `access-list`*access-list-number* {`deny`|`permit`} *protocol source source-wildcard destination destination-wildcard*
	Example:
	`Device(config)# access list 21 permit 194.1.22.0`
	or
	`Device(config)# access list 21 permit ip 194.1.22.0 10.1.1.10 194.3.44.0 10.1.1.10`
		Creates an IP standard access list, repeating the command as many times as necessary.
		or
		Creates an IP extended access list, repeating the command as many times as necessary.
		• *access-list-number*—Enters the same number created in Step 2. The range is 1 to 99 for standard access lists and 100 to 199 for extended lists.
		• **deny**—Denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched.
		• *protocol*—Enters **ip** as the protocol name.
		• *source*—Enters the number of the network or host from which the packet is being sent.
		• *source-wildcard*—Enters the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore.
		• *destination*—Enters the number of the network or host to which the packet is being sent.
		• *destination-wildcard*—Enters the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore.
		Recall that the access list is always terminated by an implicit deny statement for everything.

Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Filtering Source-Active Request Messages

By default, only Device that are caching SA information can respond to SA requests. By default, such a Device honors all SA request messages from its MSDP peers and supplies the IP addresses of the active sources.

However, you can configure the Device to ignore all SA requests from an MSDP peer. You can also honor only those SA request messages from a peer for groups described by a standard access list. If the groups in the access list pass, SA request messages are accepted. All other such messages from the peer for other groups are ignored.

To return to the default setting, use the **no ip msdp filter-sa-request** {*ip-address*| *name*} global configuration command.

Follow these steps to configure one of these options:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	Use one of the following: • `ip msdp filter-sa-request` {*ip-addressname*} • `ip msdp filter-sa-request` {*ip-addressname*} `list` *access-list-number* **Example:** `Device(config)# ip msdp filter sa-request 171.69.2.2`	Filters all SA request messages from the specified MSDP peer. or Filters SA request messages from the specified MSDP peer for groups that pass the standard access list. The access list describes a multicast group address. The range for the access-list-number is 1 to 99.
Step 4	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] **Example:** `Device(config)# access-list 1 permit 192.4.22.0 0.0.0.255`	Creates an IP standard access list, repeating the command as many times as necessary. • For *access-list-number*, the range is 1 to 99. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *source*, enter the number of the network or host from which the packet is being sent.

	Command or Action	Purpose
		• (Optional) For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Source Information that Your Switch Forwards

By default, the Device forwards all SA messages it receives to all its MSDP peers. However, you can prevent outgoing messages from being forwarded to a peer by using a filter or by setting a time-to-live (TTL) value.

Using a Filter

By creating a filter, you can perform one of these actions:

- Filter all source/group pairs
- Specify an IP extended access list to pass only certain source/group pairs
- Filter based on match criteria in a route map

Follow these steps to apply a filter:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.

Step 3	Use one of the following:
	• **ip msdp sa-filter out** {*ip-address name*}
	• **ip msdp sa-filter out** {*ip-address name*} `list` *access-list-number*
	• **ip msdp sa-filter out** {*ip-address name*} `route-map` *map-tag*
	Example:
	`Device(config)# ip msdp sa-filter out switch.cisco.com`
	or
	`Device(config)# ip msdp sa-filter out list 100`
	or
	`Device(config)# ip msdp sa-filter out switch.cisco.com route-map 22`
		• Filters all SA messages to the specified MSDP peer.
		• Passes only those SA messages that pass the IP extended access list to the specified peer. The range for the extended *access-list-number* is 100 to 199.
		If both the **list** and the **route-map** keywords are used, all conditions must be true to pass any (S,G) pair in outgoing SA messages.
		• Passes only those SA messages that meet the match criteria in the route map *map-tag* to the specified MSDP peer.
		If all match criteria are true, a **permit** from the route map passes routes through the filter. A **deny** filters routes.

Step 4	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* **Example:** `Device(config)# access list 100 permit ip 194.1.22.0 10.1.1.10 194.3.44.0 10.1.1.10`	(Optional) Creates an IP extended access list, repeating the command as many times as necessary. • For *access-list-number*, enter the number specified in Step 2. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *protocol*, enter **ip** as the protocol name. • For *source*, enter the number of the network or host from which the packet is being sent.

	Command or Action	Purpose
		• For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. • For *destination*, enter the number of the network or host to which the packet is being sent. • For *destination-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Using TTL to Limit the Multicast Data Sent in SA Messages

You can use a TTL value to control what data is encapsulated in the first SA message for every source. Only multicast packets with an IP-header TTL greater than or equal to the *ttl* argument are sent to the specified MSDP peer. For example, you can limit internal traffic to a TTL of 8. If you want other groups to go to external locations, you must send those packets with a TTL greater than 8.

Follow these steps to establish a TTL threshold:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp ttl-threshold** {*ip-address* | *name*} *ttl* **Example:** `Device(config)# ip msdp ttl-threshold switch.cisco.com 0`	Limits which multicast data is encapsulated in the first SA message to the specified MSDP peer. • For *ip-address* | *name*, enter the IP address or name of the MSDP peer to which the TTL limitation applies. • For *ttl*, enter the TTL value. The default is 0, which means all multicast data packets are forwarded to the peer until the TTL is exhausted. The range is 0 to 255.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Source Information that Your Switch Receives

By default, the Device receives all SA messages that its MSDP RPF peers send to it. However, you can control the source information that you receive from MSDP peers by filtering incoming SA messages. In other words, you can configure the Device to not accept them.

You can perform one of these actions:

- Filter all incoming SA messages from an MSDP peer
- Specify an IP extended access list to pass certain source/group pairs
- Filter based on match criteria in a route map

Follow these steps to apply a filter:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.

Step 3	Use one of the following:
	• **ip msdp sa-filter in** {*ip-address name*}
	• **ip msdp sa-filter in** {*ip-address name*} `list` *access-list-number*
	• **ip msdp sa-filter in** {*ip-address name*} `route-map` *map-tag*
	Example:
	`Device(config)# ip msdp sa-filter in switch.cisco.com`
	or
	`Device(config)# ip msdp sa-filter in list 100`
	or
	`Device(config)# ip msdp sa-filter in switch.cisco.com route-map 22`
		• Filters all SA messages to the specified MSDP peer.
		• Passes only those SA messages from the specified peer that pass the IP extended access list. The range for the extended *access-list-number* is 100 to 199.
		If both the **list** and the **route-map** keywords are used, all conditions must be true to pass any (S,G) pair in outgoing SA messages.
		• Passes only those SA messages from the specified MSDP peer that meet the match criteria in the route map *map-tag*.
		If all match criteria are true, a **permit** from the route map passes routes through the filter. A **deny** filters routes.

Step 4	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* **Example:** `Device(config)# access list 100 permit ip 194.1.22.0 10.1.1.10 194.3.44.0 10.1.1.10`	(Optional) Creates an IP extended access list, repeating the command as many times as necessary. • *access-list-number*, enter the number specified in Step 2. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *protocol*, enter **ip** as the protocol name.

	Command or Action	Purpose
		• For *source*, enter the number of the network or host from which the packet is being sent. • For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. • For *destination*, enter the number of the network or host to which the packet is being sent. • For *destination-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring an MSDP Mesh Group

An MSDP mesh group is a group of MSDP speakers that have fully meshed MSDP connectivity among one another. Any SA messages received from a peer in a mesh group are not forwarded to other peers in the same mesh group. Thus, you reduce SA message flooding and simplify peer-RPF flooding. Use the **ip msdp mesh-group** global configuration command when there are multiple RPs within a domain. It is especially used to send SA messages across a domain. You can configure multiple mesh groups (with different names) in a single Device.

Follow these steps to create a mesh group:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp mesh-group** *name* {*ip-address* | *name*} **Example:** `Device(config)# ip msdp mesh-group 2 switch.cisco.com`	Configures an MSDP mesh group, and specifies the MSDP peer belonging to that mesh group. By default, the MSDP peers do not belong to a mesh group. • For *name*, enter the name of the mesh group. • For *ip-address* | *name*, enter the IP address or name of the MSDP peer to be a member of the mesh group. Repeat this procedure on each MSDP peer in the group.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Shutting Down an MSDP Peer

If you want to configure many MSDP commands for the same peer and you do not want the peer to become active, you can shut down the peer, configure it, and later bring it up. When a peer is shut down, the TCP connection is terminated and is not restarted. You can also shut down an MSDP session without losing configuration information for the peer.

Follow these steps to shut down a peer:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp shutdown** {*peer-name* | *peer address*} **Example:** `Device(config)# ip msdp shutdown switch.cisco.com`	Shuts down the specified MSDP peer without losing configuration information. For *peer-name* | *peer address*, enter the IP address or name of the MSDP peer to shut down.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Including a Bordering PIM Dense-Mode Region in MSDP

You can configure MSDP on a Device that borders a PIM sparse-mode region with a dense-mode region. By default, active sources in the dense-mode region do not participate in MSDP.

Note We do not recommend using the **ip msdp border sa-address** global configuration command. It is better to configure the border router in the sparse-mode domain to proxy-register sources in the dense-mode domain to the RP of the sparse-mode domain and have the sparse-mode domain use standard MSDP procedures to advertise these sources.

The **ip msdp originator-id** global configuration command also identifies an interface to be used as the RP address. If both the **ip msdp border sa-address** and the **ip msdp originator-id** global configuration commands are configured, the address derived from the **ip msdp originator-id** command specifies the RP address.

Follow these steps to configure the border router to send SA messages for sources active in the dense-mode region to the MSDP peers:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip msdp border sa-address** *interface-id* **Example:** `Device(config)# ip msdp border sa-address 0/1`	Configures the switch on the border between a dense-mode and sparse-mode region to send SA messages about active sources in the dense-mode region. For *interface-id*, specifies the interface from which the IP address is derived and used as the RP address in SA messages. The IP address of the interface is used as the Originator-ID, which is the RP field in the SA message.
Step 4	**ip msdp redistribute** [**list** *access-list-name*] [**asn** *aspath-access-list-number*] [**route-map** *map*] **Example:** `Device(config)# ip msdp redistribute list 100`	Configures which (S,G) entries from the multicast routing table are advertised in SA messages. For more information, see the Redistributing Sources, on page 9.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring an Originating Address other than the RP Address

You can allow an MSDP speaker that originates an SA message to use the IP address of the interface as the RP address in the SA message by changing the Originator ID. You might change the Originator ID in one of these cases:

- If you configure a logical RP on multiple Device in an MSDP mesh group.
- If you have a Device that borders a PIM sparse-mode domain and a dense-mode domain. If a Device borders a dense-mode domain for a site, and sparse-mode is being used externally, you might want dense-mode sources to be known to the outside world. Because this Device is not an RP, it would not have an RP address to use in an SA message. Therefore, this command provides the RP address by specifying the address of the interface.

If both the **ip msdp border sa-address** and the **ip msdp originator-id** global configuration commands are configured, the address derived from the **ip msdp originator-id** command specifies the address of the RP.

Follow these steps to allow an MSDP speaker that originates an SA message to use the IP address on the interface as the RP address in the SA message:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.

	Command or Action	Purpose
Step 3	**ip msdp originator-id** *interface-id* **Example:** `Device(config)# ip msdp originator-id 0/1`	Configures the RP address in SA messages to be the address of the originating device interface. For *interface-id*, specify the interface on the local Device.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring and Maintaining MSDP

Commands that monitor MSDP SA messages, peers, state, and peer status:

Table 1: Commands for Monitoring and Maintaining MSDP

Command	Purpose
debug ip msdp [*peer-address* | *name*] [**detail**] [**routes**]	Debugs an MSDP activity.
debug ip msdp resets	Debugs MSDP peer reset reasons.
show ip msdp count [*autonomous-system-number*]	Displays the number of sources and groups originated in SA messages from each autonomous system. The **ip msdp cache-sa-state** command must be configured for this command to produce any output.
show ip msdp peer [*peer-address* | *name*]	Displays detailed information about an MSDP peer.
show ip msdp sa-cache [*group-address* | *source-address* | *group-name* | *source-name*] [*autonomous-system-number*]	Displays (S,G) state learned from MSDP peers.

Command	Purpose
show ip msdp summary	Displays MSDP peer status and SA message counts.

Commands that clear MSDP connections, statistics, and SA cache entries:

Table 2: Commands for Clearing MSDP Connections, Statistics, or SA Cache Entries

Command	Purpose
clear ip msdp peer *peer-address* | *name*	Clears the TCP connection to the specified MSDP peer, resetting all MSDP message counters.
clear ip msdp statistics [*peer-address* | *name*]	Clears statistics counters for one or all the MSDP peers without resetting the sessions.
clear ip msdp sa-cache [*group-address* | *name*]	Clears the SA cache entries for all entries, all sources for a specific group, or all entries for a specific source/group pair.

Configuration Examples for Configuring MSDP

Configuring a Default MSDP Peer: Example

This example shows a partial configuration of Router A and Router C in . Each of these ISPs have more than one customer (like the customer in) who use default peering (no BGP or MBGP). In that case, they might have similar configurations. That is, they accept SAs only from a default peer if the SA is permitted by the corresponding prefix list.

Router A

```
Router(config)# ip msdp default-peer 10.1.1.1
Router(config)# ip msdp default-peer 10.1.1.1 prefix-list site-a
Router(config)# ip prefix-list site-b permit 10.0.0.0/1
```

Router C

```
Router(config)# ip msdp default-peer 10.1.1.1 prefix-list site-a
Router(config)# ip prefix-list site-b permit 10.0.0.0/1
```

Caching Source-Active State: Example

This example shows how to enable the cache state for all sources in 171.69.0.0/16 sending to groups 224.2.0.0/16:

```
Device(config)# ip msdp cache-sa-state 100
Device(config)# access-list 100 permit ip 171.69.0.0 0.0.255.255 224.2.0.0 0.0.255.255
```

Requesting Source Information from an MSDP Peer: Example

This example shows how to configure the switch to send SA request messages to the MSDP peer at 171.69.1.1:

```
Device(config)# ip msdp sa-request 171.69.1.1
```

Controlling Source Information that Your Switch Originates: Example

This example shows how to configure the switch to filter SA request messages from the MSDP peer at 171.69.2.2. SA request messages from sources on network 192.4.22.0 pass access list 1 and are accepted; all others are ignored.

```
Device(config)# ip msdp filter sa-request 171.69.2.2 list 1
Device(config)# access-list 1 permit 192.4.22.0 0.0.0.255
```

Controlling Source Information that Your Switch Forwards: Example

This example shows how to allow only (S,G) pairs that pass access list 100 to be forwarded in an SA message to the peer named *switch.cisco.com*:

```
Device(config)# ip msdp peer switch.cisco.com connect-source gigabitethernet1/0/1
Device(config)# ip msdp sa-filter out switch.cisco.com list 100
Device(config)# access-list 100 permit ip 171.69.0.0 0.0.255.255 224.20 0 0.0.255.255
```

Controlling Source Information that Your Switch Receives: Example

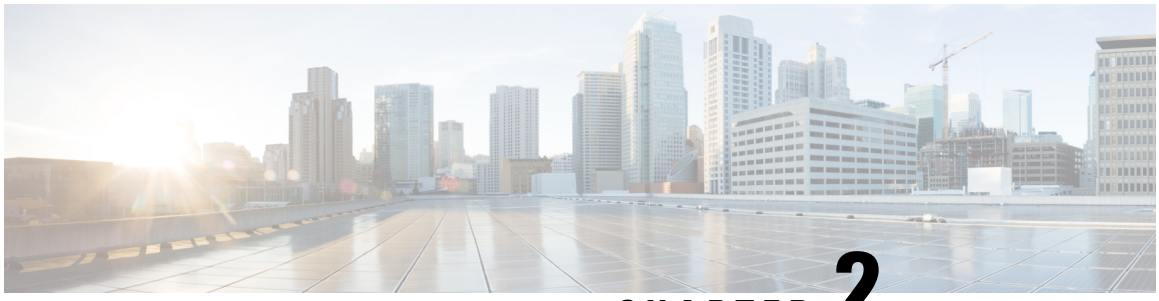
This example shows how to filter all SA messages from the peer named *switch.cisco.com*:

```
Device(config)# ip msdp peer switch.cisco.com connect-source gigabitethernet1/0/1
Device(config)# ip msdp sa-filter in switch.cisco.com
```

Feature Information for Multicast Source Discovery Protocol

Table 3: Feature Information for Multicast Source Discovery Protocol

Feature Name	Release	Feature Information
Multicast Source Discovery Protocol	Cisco IOS XE Everest 16.6.1	This feature was introduced



CHAPTER 2

Configuring IP Unicast Routing

Restrictions for IP Unicast Routing

- The switch does not support tunnel interfaces for unicast routed traffic.
- Subnetwork Access Protocol (SNAP) address resolution is not supported on this device.

Information About Configuring IP Unicast Routing

This module describes how to configure IP Version 4 (IPv4) unicast routing on the switch.

A switch stack operates and appears as a single router to the rest of the routers in the network. Basic routing functions like static routing are available with .

Note In addition to IPv4 traffic, you can also enable IP Version 6 (IPv6) unicast routing and configure interfaces to forward IPv6 traffic .

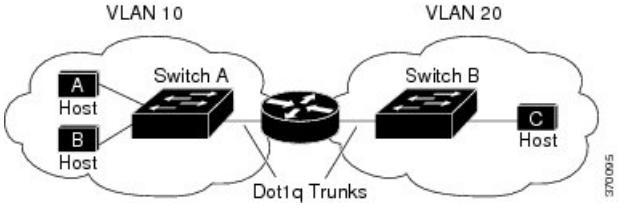
Information About IP Routing

In some network environments, VLANs are associated with individual networks or subnetworks. In an IP network, each subnetwork is mapped to an individual VLAN. Configuring VLANs helps control the size of

the broadcast domain and keeps local traffic local. However, network devices in different VLANs cannot communicate with one another without a Layer 3 device (router) to route traffic between the VLAN, referred to as inter-VLAN routing. You configure one or more routers to route traffic to the appropriate destination VLAN.

Figure 2: Routing Topology Example

This figure shows a basic routing topology. Switch A is in VLAN 10, and Switch B is in VLAN 20. The router has an interface in each VLAN.



When Host A in VLAN 10 needs to communicate with Host B in VLAN 10, it sends a packet addressed to that host. Switch A forwards the packet directly to Host B, without sending it to the router.

When Host A sends a packet to Host C in VLAN 20, Switch A forwards the packet to the router, which receives the traffic on the VLAN 10 interface. The router checks the routing table, finds the correct outgoing interface, and forwards the packet on the VLAN 20 interface to Switch B. Switch B receives the packet and forwards it to Host C.

Types of Routing

Routers and Layer 3 switches can route packets in these ways:

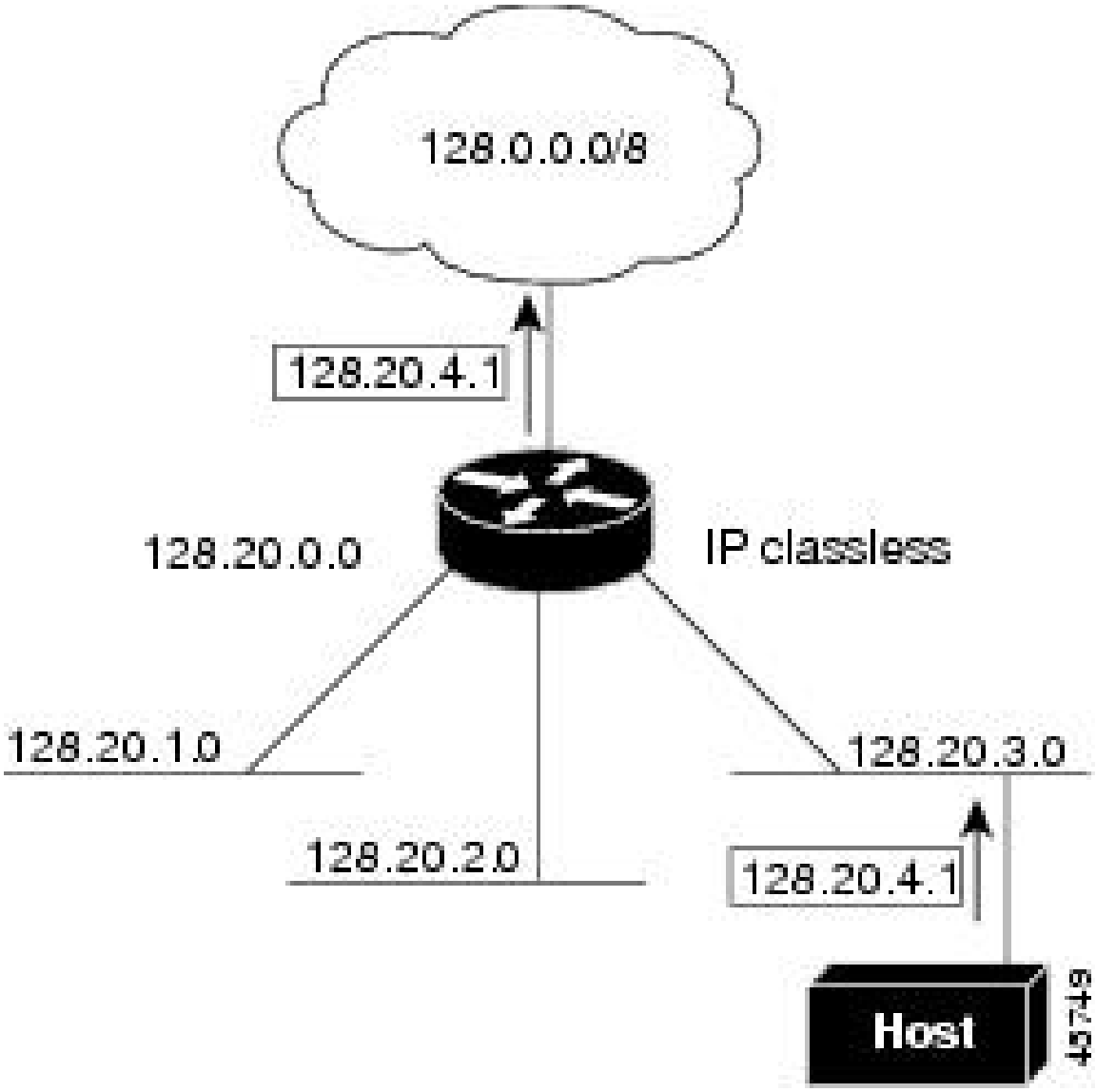
- By using default routing
- By using preprogrammed static routes for the traffic
- By dynamically calculating routes by using a routing protocol

Classless Routing

By default, classless routing behavior is enabled on the Device when it is configured to route. With classless routing, if a router receives packets for a subnet of a network with no default route, the router forwards the packet to the best supernet route. A supernet consists of contiguous blocks of Class C address spaces used to simulate a single, larger address space and is designed to relieve the pressure on the rapidly depleting Class B address space.

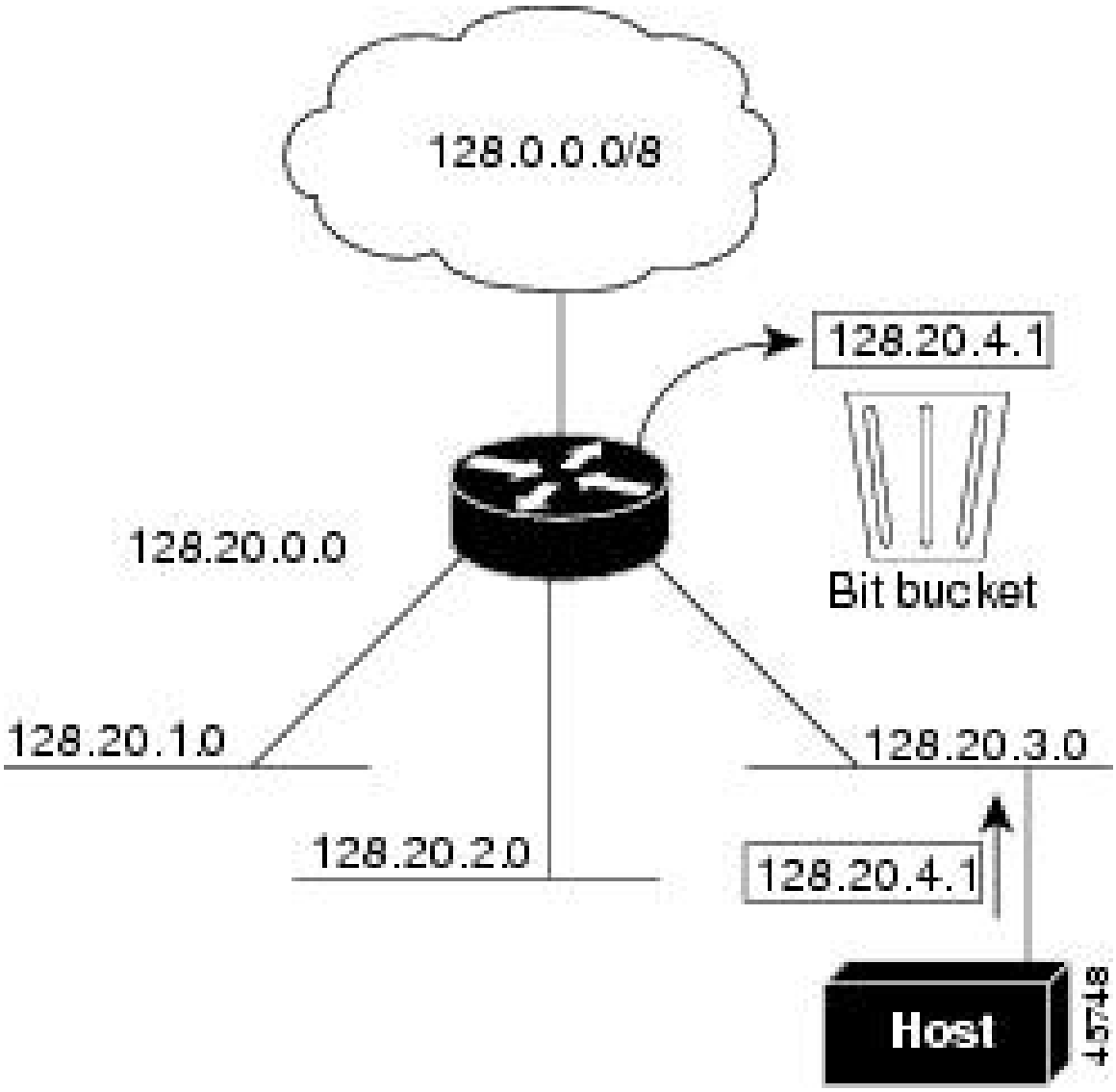
In the figure, classless routing is enabled. When the host sends a packet to 120.20.4.1, instead of discarding the packet, the router forwards it to the best supernet route. If you disable classless routing and a router receives packets destined for a subnet of a network with no network default route, the router discards the packet.

Figure 3: IP Classless Routing



In the figure , the router in network 128.20.0.0 is connected to subnets 128.20.1.0, 128.20.2.0, and 128.20.3.0. If the host sends a packet to 120.20.4.1, because there is no network default route, the router discards the packet.

Figure 4: No IP Classless Routing



To prevent the Device from forwarding packets destined for unrecognized subnets to the best supernet route possible, you can disable classless routing behavior.

Address Resolution

You can control interface-specific handling of IP by using address resolution. A device using IP can have both a local address or MAC address, which uniquely defines the device on its local segment or LAN, and a network address, which identifies the network to which the device belongs.

The local address or MAC address is known as a data link address because it is contained in the data link layer (Layer 2) section of the packet header and is read by data link (Layer 2) devices. To communicate with a device on Ethernet, the software must learn the MAC address of the device. The process of learning the MAC address from an IP address is called *address resolution*. The process of learning the IP address from the MAC address is called *reverse address resolution*.

The Device can use these forms of address resolution:

- Address Resolution Protocol (ARP) is used to associate IP address with MAC addresses. Taking an IP address as input, ARP learns the associated MAC address and then stores the IP address/MAC address association in an ARP cache for rapid retrieval. Then the IP datagram is encapsulated in a link-layer frame and sent over the network.
- Proxy ARP helps hosts with no routing tables learn the MAC addresses of hosts on other networks or subnets. If the Device (router) receives an ARP request for a host that is not on the same interface as the ARP request sender, and if the router has all of its routes to the host through other interfaces, it generates a proxy ARP packet giving its own local data link address. The host that sent the ARP request then sends its packets to the router, which forwards them to the intended host.

The Device also uses the Reverse Address Resolution Protocol (RARP), which functions the same as ARP does, except that the RARP packets request an IP address instead of a local MAC address. Using RARP requires a RARP server on the same network segment as the router interface. Use the **ip rarp-server** *address* interface configuration command to identify the server.

Proxy ARP

Proxy ARP, the most common method for learning about other routes, enables an Ethernet host with no routing information to communicate with hosts on other networks or subnets. The host assumes that all hosts are on the same local Ethernet and that they can use ARP to learn their MAC addresses. If a Device receives an ARP request for a host that is not on the same network as the sender, the Device evaluates whether it has the best route to that host. If it does, it sends an ARP reply packet with its own Ethernet MAC address, and the host that sent the request sends the packet to the Device, which forwards it to the intended host. Proxy ARP treats all networks as if they are local, and performs ARP requests for every IP address.

ICMP Router Discovery Protocol

Router discovery allows the Device to dynamically learn about routes to other networks using ICMP router discovery protocol (IRDP). IRDP allows hosts to locate routers. When operating as a client, the Device generates router discovery packets. When operating as a host, the Device receives router discovery packets. The Device can also listen to Routing Information Protocol (RIP) routing updates and use this information to infer locations of routers. The Device does not actually store the routing tables sent by routing devices; it merely keeps track of which systems are sending the data. The advantage of using IRDP is that it allows each router to specify both a priority and the time after which a device is assumed to be down if no further packets are received.

Each device discovered becomes a candidate for the default router, and a new highest-priority router is selected when a higher priority router is discovered, when the current default router is declared down, or when a TCP connection is about to time out because of excessive retransmissions.

UDP Broadcast Packets and Protocols

User Datagram Protocol (UDP) is an IP host-to-host layer protocol, as is TCP. UDP provides a low-overhead, connectionless session between two end systems and does not provide for acknowledgment of received datagrams. Network hosts occasionally use UDP broadcasts to find address, configuration, and name information. If such a host is on a network segment that does not include a server, UDP broadcasts are normally not forwarded. You can remedy this situation by configuring an interface on a router to forward certain classes of broadcasts to a helper address. You can use more than one helper address per interface.

You can specify a UDP destination port to control which UDP services are forwarded. You can specify multiple UDP protocols. You can also specify the Network Disk (ND) protocol, which is used by older diskless Sun workstations and the network security protocol SDNS.

By default, both UDP and ND forwarding are enabled if a helper address has been defined for an interface.

Broadcast Packet Handling

After configuring an IP interface address, you can enable routing and configure one or more routing protocols, or you can configure the way the Device responds to network broadcasts. A broadcast is a data packet destined for all hosts on a physical network. The Device supports two kinds of broadcasting:

- A directed broadcast packet is sent to a specific network or series of networks. A directed broadcast address includes the network or subnet fields.
- A flooded broadcast packet is sent to every network.

Note You can also limit broadcast, unicast, and multicast traffic on Layer 2 interfaces by using the **storm-control** interface configuration command to set traffic suppression levels.

Routers provide some protection from broadcast storms by limiting their extent to the local cable. Bridges (including intelligent bridges), because they are Layer 2 devices, forward broadcasts to all network segments, thus propagating broadcast storms. The best solution to the broadcast storm problem is to use a single broadcast address scheme on a network. In most modern IP implementations, you can set the address to be used as the broadcast address. Many implementations, including the one in the Device, support several addressing schemes for forwarding broadcast messages.

IP Broadcast Flooding

You can allow IP broadcasts to be flooded throughout your internetwork in a controlled fashion by using the database created by the bridging STP. Using this feature also prevents loops. To support this capability, bridging must be configured on each interface that is to participate in the flooding. If bridging is not configured on an interface, it still can receive broadcasts. However, the interface never forwards broadcasts it receives, and the router never uses that interface to send broadcasts received on a different interface.

Packets that are forwarded to a single network address using the IP helper-address mechanism can be flooded. Only one copy of the packet is sent on each network segment.

To be considered for flooding, packets must meet these criteria. (Note that these are the same conditions used to consider packet forwarding using IP helper addresses.)

- The packet must be a MAC-level broadcast.
- The packet must be an IP-level broadcast.
- The packet must be a TFTP, DNS, Time, NetBIOS, ND, or BOOTP packet, or a UDP specified by the **ip forward-protocol udp** global configuration command.
- The time-to-live (TTL) value of the packet must be at least two.

A flooded UDP datagram is given the destination address specified with the **ip broadcast-address** interface configuration command on the output interface. The destination address can be set to any address. Thus, the destination address might change as the datagram propagates through the network. The source address is never changed. The TTL value is decremented.

When a flooded UDP datagram is sent out an interface (and the destination address possibly changed), the datagram is handed to the normal IP output routines and is, therefore, subject to access lists, if they are present on the output interface.

In the Device, the majority of packets are forwarded in hardware; most packets do not go through the Device CPU. For those packets that do go to the CPU, you can speed up spanning tree-based UDP flooding by a factor of about four to five times by using turbo-flooding. This feature is supported over Ethernet interfaces configured for ARP encapsulation.

How to Configure IP Routing

By default, IP routing is disabled on the Device, and you must enable it before routing can take place.

In the following procedures, the specified interface must be one of these Layer 3 interfaces:

- A routed port: a physical port configured as a Layer 3 port by using the **no switchport** interface configuration command.
- A switch virtual interface (SVI): a VLAN interface created by using the **interface vlan** *vlan_id* global configuration command and by default a Layer 3 interface.
- An EtherChannel port channel in Layer 3 mode: a port-channel logical interface created by using the **interface port-channel** *port-channel-number* global configuration command and binding the Ethernet interface into the channel group.

All Layer 3 interfaces on which routing will occur must have IP addresses assigned to them.

Note A Layer 3 switch can have an IP address assigned to each routed port and SVI.

Configuring routing consists of several main procedures:

- To support VLAN interfaces, create and configure VLANs on the Device or switch stack, and assign VLAN membership to Layer 2 interfaces. For more information, see the "Configuring VLANs" chapter.
- Configure Layer 3 interfaces.
- Enable IP routing on the switch.
- Assign IP addresses to the Layer 3 interfaces.
- Enable selected routing protocols on the switch.
- Configure routing protocol parameters (optional).

How to Configure IP Addressing

A required task for configuring IP routing is to assign IP addresses to Layer 3 network interfaces to enable the interfaces and allow communication with the hosts on those interfaces that use IP. The following sections describe how to configure various IP addressing features. Assigning IP addresses to the interface is required; the other procedures are optional.

- Default Addressing Configuration
- Assigning IP Addresses to Network Interfaces
- Configuring Address Resolution Methods
- Routing Assistance When IP Routing is Disabled
- Configuring Broadcast Packet Handling
- Monitoring and Maintaining IP Addressing

Default IP Addressing Configuration

Table 4: Default Addressing Configuration

Feature	Default Setting
IP address	None defined.
ARP	No permanent entries in the Address Resolution Protocol (ARP) cache. Encapsulation: Standard Ethernet-style ARP. Timeout: 14400 seconds (4 hours).
IP broadcast address	255.255.255.255 (all ones).
IP classless routing	Enabled.
IP default gateway	Disabled.
IP directed broadcast	Disabled (all IP directed broadcasts are dropped).
IP domain	Domain list: No domain names defined. Domain lookup: Enabled. Domain name: Enabled.
IP forward-protocol	If a helper address is defined or User Datagram Protocol (UDP) flooding is configured, UDP f is enabled on default ports. Any-local-broadcast: Disabled. Spanning Tree Protocol (STP): Disabled. Turbo-flood: Disabled.

Feature	Default Setting
IP helper address	Disabled.
IP host	Disabled.
IRDP	Disabled. Defaults when enabled: • Broadcast IRDP advertisements. • Maximum interval between advertisements: 600 seconds. • Minimum interval between advertisements: 0.75 times max interval • Preference: 0.
IP proxy ARP	Enabled.
IP routing	Disabled.
IP subnet-zero	Disabled.

Assigning IP Addresses to Network Interfaces

An IP address identifies a location to which IP packets can be sent. Some IP addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. RFC 1166, “Internet Numbers,” contains the official description of IP addresses.

An interface can have one primary IP address. A mask identifies the bits that denote the network number in an IP address. When you use the mask to subnet a network, the mask is referred to as a subnet mask. To receive an assigned network number, contact your Internet service provider.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:**	Enters interface configuration mode, and specifies the Layer 3 interface to configure.

	Command or Action	Purpose
	`Device(config)# interface gigabitethernet 1/0/1`	
Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Removes the interface from Layer 2 configuration mode (if it is a physical interface).
Step 5	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)# ip address 10.1.5.1 255.255.255.0`	Configures the IP address and IP subnet mask.
Step 6	**no shutdown** **Example:** `Device(config-if)# no shutdown`	Enables the physical interface.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show ip route** **Example:** `Device# show ip route`	Verifies your entries.
Step 9	**show ip interface** [*interface-id*] **Example:** `Device# show ip interface gigabitethernet 1/0/1`	Verifies your entries.
Step 10	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Using Subnet Zero

Subnetting with a subnet address of zero is strongly discouraged because of the problems that can arise if a network and a subnet have the same addresses. For example, if network 131.108.0.0 is subnetted as 255.255.255.0, subnet zero would be written as 131.108.0.0, which is the same as the network address.

You can use the all ones subnet (131.108.255.0) and even though it is discouraged, you can enable the use of subnet zero if you need the entire subnet space for your IP address.

Use the **no ip subnet-zero** global configuration command to restore the default and disable the use of subnet zero.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip subnet-zero** **Example:** `Device(config)# ip subnet-zero`	Enables the use of subnet zero for interface addresses and routing updates.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Disabling Classless Routing

To prevent the Device from forwarding packets destined for unrecognized subnets to the best supernet route possible, you can disable classless routing behavior.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**no ip classless** **Example:** `Device(config)#no ip classless`	Disables classless routing behavior.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Address Resolution Methods

You can perform the following tasks to configure address resolution.

Defining a Static ARP Cache

ARP and other address resolution protocols provide dynamic mapping between IP addresses and MAC addresses. Because most hosts support dynamic address resolution, you usually do not need to specify static ARP cache entries. If you must define a static ARP cache entry, you can do so globally, which installs a permanent entry in the ARP cache that the Device uses to translate IP addresses into MAC addresses. Optionally, you can also specify that the Device respond to ARP requests as if it were the owner of the specified IP address. If you do not want the ARP entry to be permanent, you can specify a timeout period for the ARP entry.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**arp** *ip-address hardware-address type* **Example:** `Device(config)# ip 10.1.5.1 c2f3.220a.12f4 arpa`	Associates an IP address with a MAC (hardware) address in the ARP cache, and specifies encapsulation type as one of these: • **arpa**—ARP encapsulation for Ethernet interfaces • **sap**—HP's ARP type
Step 4	**arp** *ip-address hardware-address type* [**alias**] **Example:** `Device(config)# ip 10.1.5.3 d7f3.220d.12f5 arpa alias`	(Optional) Specifies that the switch respond to ARP requests as if it were the owner of the specified IP address.
Step 5	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 6	**arp timeout** *seconds* **Example:** `Device(config-if)# arp timeout 20000`	(Optional) Sets the length of time an ARP cache entry will stay in the cache. The recommended value of ARP timeout is 4 hours which is also the default setting. However, if your network experiences regular updates to ARP cache entries, consider reducing the timeout. Be aware that decreasing the ARP timeout can result in increased network traffic. Setting the ARP timeout to 60 seconds or less is

	Command or Action	Purpose
		generally not recommended as it may cause network disruptions.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show interfaces** [*interface-id*] **Example:** `Device# show interfaces gigabitethernet 1/0/1`	Verifies the type of ARP and the timeout value used on all interfaces or a specific interface.
Step 9	**show arp** **Example:** `Device# show arp`	Views the contents of the ARP cache.
Step 10	**show ip arp** **Example:** `Device# show ip arp`	Views the contents of the ARP cache.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Setting ARP Encapsulation

By default, Ethernet ARP encapsulation (represented by the **arpa** keyword) is enabled on an IP interface.

To disable an encapsulation type, use the **no arp arpa** interface configuration command.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters the global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/2`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**arp arpa** **Example:** `Device(config-if)# arp arpa`	Specifies the ARP encapsulation method:
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show interfaces** [*interface-id*] **Example:** `Device# show interfaces`	Verifies ARP encapsulation configuration on all interfaces or the specified interface.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Enabling Proxy ARP

By default, the Device uses proxy ARP to help hosts learn MAC addresses of hosts on other networks or subnets.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters the global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/2`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip proxy-arp** **Example:** `Device(config-if)# ip proxy-arp`	Enables proxy ARP on the interface.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ip interface** [*interface-id*] **Example:** `Device# show ip interface gigabitethernet 1/0/2`	Verifies the configuration on the interface or all interfaces.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Routing Assistance When IP Routing is Disabled

These mechanisms allow the Device to learn about routes to other networks when it does not have IP routing enabled:

- Proxy ARP
- Default Gateway
- ICMP Router Discovery Protocol (IRDP)

Proxy ARP

Proxy ARP is enabled by default. To enable it after it has been disabled, see the “Enabling Proxy ARP” section. Proxy ARP works as long as other routers support it.

Default Gateway

Another method for locating routes is to define a default router or default gateway. All non-local packets are sent to this router, which either routes them appropriately or sends an IP Control Message Protocol (ICMP) redirect message back, defining which local router the host should use. The Device caches the redirect messages and forwards each packet as efficiently as possible. A limitation of this method is that there is no means of detecting when the default router has gone down or is unavailable.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip default-gateway** *ip-address* **Example:** `Device(config)# ip default gateway 10.1.5.1`	Sets up a default gateway (router).
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show ip redirects** **Example:** `Device# show ip redirects`	Displays the address of the default gateway router to verify the setting.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

ICMP Router Discovery Protocol (IRDP)

The only required task for IRDP routing on an interface is to enable IRDP processing on that interface. When enabled, the default parameters apply.

You can optionally change any of these parameters. If you change the **maxadvertinterval** value, the **holdtime** and **minadvertinterval** values also change, so it is important to first change the **maxadvertinterval** value, before manually changing either the **holdtime** or **minadvertinterval** values.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip irdp** **Example:** `Device(config-if)# ip irdp`	Enables IRDP processing on the interface.
Step 5	**ip irdp multicast** **Example:** `Device(config-if)# ip irdp multicast`	(Optional) Sends IRDP advertisements to the multicast address (224.0.0.1) instead of IP broadcasts. **Note** This command allows for compatibility with Sun Microsystems Solaris, which requires IRDP packets to be sent out as multicasts. Many implementations cannot receive these multicasts; ensure end-host ability before using this command.
Step 6	**ip irdp holdtime** *seconds* **Example:** `Device(config-if)# ip irdp holdtime 1000`	(Optional) Sets the IRDP period for which advertisements are valid. The default is three times the **maxadvertinterval** value. It must be greater than **maxadvertinterval** and cannot be greater than 9000 seconds. If you change the **maxadvertinterval** value, this value also changes.
Step 7	**ip irdp maxadvertinterval** *seconds* **Example:** `Device(config-if)# ip irdp maxadvertinterval 650`	(Optional) Sets the IRDP maximum interval between advertisements. The default is 600 seconds.

	Command or Action	Purpose
Step 8	**ip irdp minadvertinterval** *seconds* **Example:** `Device(config-if)# ip irdp minadvertinterval 500`	(Optional) Sets the IRDP minimum interval between advertisements. The default is 0.75 times the **maxadvertinterval**. If you change the **maxadvertinterval**, this value changes to the new default (0.75 of **maxadvertinterval**).
Step 9	**ip irdp preference** *number* **Example:** `Device(config-if)# ip irdp preference 2`	(Optional) Sets a device IRDP preference level. The allowed range is –231 to 231. The default is 0. A higher value increases the router preference level.
Step 10	**ip irdp address** *address* [*number*] **Example:** `Device(config-if)# ip irdp address 10.1.10.10`	(Optional) Specifies an IRDP address and preference to proxy-advertise.
Step 11	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 12	**show ip irdp** **Example:** `Device# show ip irdp`	Verifies settings by displaying IRDP values.
Step 13	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Broadcast Packet Handling

Perform the tasks in these sections to enable these schemes:

- Enabling Directed Broadcast-to-Physical Broadcast Translation
- Forwarding UDP Broadcast Packets and Protocols
- Establishing an IP Broadcast Address
- Flooding IP Broadcasts

Enabling Directed Broadcast-to-Physical Broadcast Translation

By default, IP directed broadcasts are dropped; they are not forwarded. Dropping IP-directed broadcasts makes routers less susceptible to denial-of-service attacks.

You can enable forwarding of IP-directed broadcasts on an interface where the broadcast becomes a physical (MAC-layer) broadcast. Only those protocols configured by using the **ip forward-protocol** global configuration command are forwarded.

You can specify an access list to control which broadcasts are forwarded. When an access list is specified, only those IP packets permitted by the access list are eligible to be translated from directed broadcasts to physical broadcasts. For more information on access lists, see the “Configuring ACLs" chapter in the Security section.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/2`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip directed-broadcast** [*access-list-number*] **Example:** `Device(config-if)# ip directed-broadcast 103`	Enables directed broadcast-to-physical broadcast translation on the interface. You can include an access list to control which broadcasts are forwarded. When an access list, only IP packets permitted by the access list can be translated.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 6	**ip forward-protocol** {**udp** [*port*] | **nd** | **sdns**} **Example:** `Device(config)# ip forward-protocol nd`	Specifies which protocols and ports the router forwards when forwarding broadcast packets. • **udp**—Forward UPD datagrams. port: (Optional) Destination port that controls which UDP services are forwarded. • **nd**—Forward ND datagrams. • **sdns**—Forward SDNS datagrams

	Command or Action	Purpose
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show ip interface** [*interface-id*] **Example:** `Device# show ip interface`	Verifies the configuration on the interface or all interfaces
Step 9	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 10	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Forwarding UDP Broadcast Packets and Protocols

If you do not specify any UDP ports when you configure the forwarding of UDP broadcasts, you are configuring the router to act as a BOOTP forwarding agent. BOOTP packets carry DHCP information.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.

	Command or Action	Purpose
Step 4	**ip helper-address** *address* **Example:** `Device(config-if)# ip helper address 10.1.10.1`	Enables forwarding and specifies the destination address for forwarding UDP broadcast packets, including BOOTP.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 6	**ip forward-protocol** {**udp** [*port*] | **nd** | **sdns**} **Example:** `Device(config)# ip forward-protocol sdns`	Specifies which protocols the router forwards when forwarding broadcast packets.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show ip interface** [*interface-id*] **Example:** `Device# show ip interface gigabitethernet 1/0/1`	Verifies the configuration on the interface or all interfaces.
Step 9	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 10	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Establishing an IP Broadcast Address

The most popular IP broadcast address (and the default) is an address consisting of all ones (255.255.255.255). However, the Device can be configured to generate any form of IP broadcast address.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip broadcast-address** *ip-address* **Example:** `Device(config-if)# ip broadcast-address 128.1.255.255`	Enters a broadcast address different from the default, for example 128.1.255.255.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ip interface** [*interface-id*] **Example:** `Device# show ip interface`	Verifies the broadcast address on the interface or all interfaces.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Flooding IP Broadcasts

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip forward-protocol spanning-tree** **Example:** `Device(config)# ip forward-protocol spanning-tree`	Uses the bridging spanning-tree database to flood UDP datagrams.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.
Step 7	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 8	**ip forward-protocol turbo-flood** **Example:**	Uses the spanning-tree database to speed up flooding of UDP datagrams.

	Command or Action	Purpose
	`Device(config)# ip forward-protocol turbo-flood`	
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 10	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring and Maintaining IP Addressing

When the contents of a particular cache, table, or database have become or are suspected to be invalid, you can remove all its contents by using the **clear** privileged EXEC commands. The Table lists the commands for clearing contents.

Table 5: Commands to Clear Caches, Tables, and Databases

clear arp-cache	Clears the IP ARP cache and the fast-switching cache.
clear host {*name* | *****}	Removes one or all entries from the hostname and the address
clear ip route {**network** [*mask*] | *****}	Removes one or more routes from the IP routing table.

You can display specific statistics, such as the contents of IP routing tables, caches, and databases; the reachability of nodes; and the routing path that packets are taking through the network. The Table lists the privileged EXEC commands for displaying IP statistics.

Table 6: Commands to Display Caches, Tables, and Databases

show arp	Displays the entries in the ARP table.
show hosts	Displays the default domain name, style of lookup service, name ser the cached list of hostnames and addresses.
show ip aliases	Displays IP addresses mapped to TCP ports (aliases).
show ip arp	Displays the IP ARP cache.

show ip interface [*interface-id*]	Displays the IP status of interfaces.
show ip irdp	Displays IRDP values.
show ip masks *address*	Displays the masks used for network addresses and the number of subn each mask.
show ip redirects	Displays the address of a default gateway.
show ip route [*address* [*mask*]] | [*protocol*]	Displays the current state of the routing table.
show ip route summary	Displays the current state of the routing table in summary form.

How to Configure IP Unicast Routing

Enabling IP Unicast Routing

By default, the Device is in Layer 2 switching mode and IP routing is disabled. To use the Layer 3 capabilities of the Device, you must enable IP routing.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Example of Enabling IP Routing

This example shows how to enable IP routing :

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# ip routing

Device(config-router)# end
```

What to Do Next

You can now set up parameters for the selected routing protocols as described in these sections:

- RIP
- OSPF,
- EIGRP
- BGP
- Unicast Reverse Path Forwarding
- Protocol-Independent Features (optional)

Monitoring and Maintaining the IP Network

You can remove all contents of a particular cache, table, or database. You can also display specific statistics.

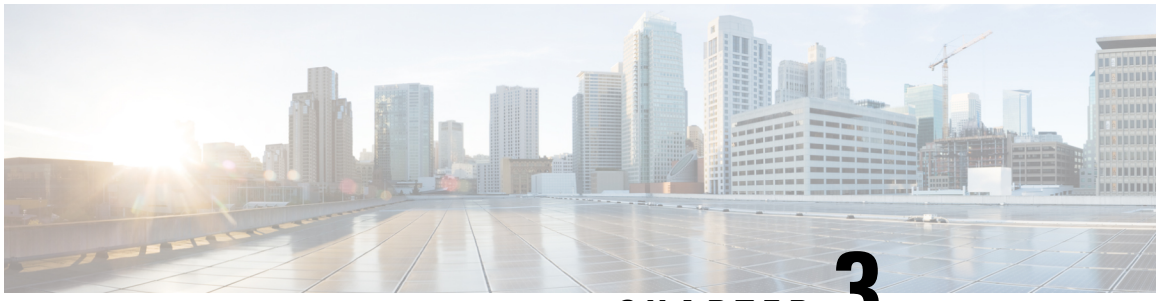
Table 7: Commands to Clear IP Routes or Display Route Status

Command	Purpose
show ip route summary	Displays the current state of the routing table in summary

Feature Information for IP Unicast Routing

Table 8: Feature Information for IP Unicast Routing

Feature Name	Release	Feature Information
IP Unicast Routing	Cisco IOS XE Everest 16.6.1	This feature was introduced



CHAPTER 3

Configuring RIP

Information About RIP

The Routing Information Protocol (RIP) is an interior gateway protocol (IGP) created for use in small, homogeneous networks. It is a distance-vector routing protocol that uses broadcast User Datagram Protocol (UDP) data packets to exchange routing information. The protocol is documented in RFC 1058. You can find detailed information about RIP in *IP Routing Fundamentals,* published by Cisco Press.

Note RIP is supported in the Network Essentials feature set.

Using RIP, the Device sends routing information updates (advertisements) every 30 seconds. If a router does not receive an update from another router for 180 seconds or more, it marks the routes served by that router as unusable. If there is still no update after 240 seconds, the router removes all routing table entries for the non-updating router.

RIP uses hop counts to rate the value of different routes. The hop count is the number of routers that can be traversed in a route. A directly connected network has a hop count of zero; a network with a hop count of 16 is unreachable. This small range (0 to 15) makes RIP unsuitable for large networks.

If the router has a default network path, RIP advertises a route that links the router to the pseudonetwork 0.0.0.0. The 0.0.0.0 network does not exist; it is treated by RIP as a network to implement the default routing feature. The Device advertises the default network if a default was learned by RIP or if the router has a gateway of last resort and RIP is configured with a default metric. RIP sends updates to the interfaces in specified networks. If an interface's network is not specified, it is not advertised in any RIP update.

Summary Addresses and Split Horizon

Routers connected to broadcast-type IP networks and using distance-vector routing protocols normally use the split-horizon mechanism to reduce the possibility of routing loops. Split horizon blocks information about

routes from being advertised by a router on any interface from which that information originated. This feature usually optimizes communication among multiple routers, especially when links are broken.

How to Configure RIP

Default RIP Configuration

Table 9: Default RIP Configuration

Feature	Default Setting
Auto summary	Enabled.
Default-information originate	Disabled.
Default metric	Built-in; automatic metric translations.
IP RIP authentication key-chain	No authentication. Authentication mode: clear text.
IP RIP triggered	Disabled
IP split horizon	Varies with media.
Neighbor	None defined.
Network	None specified.
Offset list	Disabled.
Output delay	0 milliseconds.
Timers basic	• Update: 30 seconds. • Invalid: 180 seconds. • Hold-down: 180 seconds. • Flush: 240 seconds.
Validate-update-source	Enabled.
Version	Receives RIP Version 1 and 2 packets; sends Version 1 packets.

Configuring Basic RIP Parameters

To configure RIP, you enable RIP routing for a network and optionally configure other parameters. On the Device, RIP configuration commands are ignored until you configure the network number.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing. (Required only if IP routing is disabled.)
Step 4	**router rip** **Example:** `Device(config)# router rip`	Enables a RIP routing process, and enter router configuration mode.
Step 5	**network** *network number* **Example:** `Device(config-router)# network 12.0.0.0`	Associates a network with a RIP routing process. You can specify multiple **network** commands. RIP routing updates are sent and received through interfaces only on these networks. **Note** You must configure a network number for the RIP commands to take effect.
Step 6	**neighbor** *ip-address* **Example:** `Device(config-router)# neighbor 10.2.5.1`	(Optional) Defines a neighboring router with which to exchange routing information. This step allows routing updates from RIP (normally a broadcast protocol) to reach nonbroadcast networks.
Step 7	**offset-list** [*access-list number* | *name*] {**in** | **out**} *offset* [*type number*] **Example:** `Device(config-router)# offset-list 103 in 10`	(Optional) Applies an offset list to routing metrics to increase incoming and outgoing metrics to routes learned through RIP. You can limit the offset list with an access list or an interface.
Step 8	**timers basic** *update invalid holddown flush* **Example:** `Device(config-router)# timers basic 45 360 400 300`	(Optional) Adjusts routing protocol timers. Valid ranges for all timers are 0 to 4294967295 seconds. • *update*—The time between sending routing updates. The default is 30 seconds.

	Command or Action	Purpose
		• *invalid*—The timer after which a route is declared invalid. The default is 180 seconds. • *holddown*—The time before a route is removed from the routing table. The default is 180 seconds. • *flush*—The amount of time for which routing updates are postponed. The default is 240 seconds.
Step 9	**version** {**1** | **2**} **Example:** `Device(config-router)# version 2`	(Optional) Configures the switch to receive and send only RIP Version 1 or RIP Version 2 packets. By default, the switch receives Version 1 and 2 but sends only Version 1. You can also use the interface commands **ip rip** {**send** | **receive**} **version 1** | **2** | **1 2**} to control what versions are used for sending and receiving on interfaces.
Step 10	**no auto summary** **Example:** `Device(config-router)# no auto summary`	(Optional) Disables automatic summarization. By default, the switch summarizes subprefixes when crossing classful network boundaries. Disable summarization (RIP Version 2 only) to advertise subnet and host routing information to classful network boundaries.
Step 11	**output-delay** *delay* **Example:** `Device(config-router)# output-delay 8`	(Optional) Adds interpacket delay for RIP updates sent. By default, packets in a multiple-packet RIP update have no delay added between packets. If you are sending packets to a lower-speed device, you can add an interpacket delay in the range of 8 to 50 milliseconds.
Step 12	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 13	**show ip protocols** **Example:** `Device# show ip protocols`	Verifies your entries.
Step 14	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring RIP Authentication

RIP Version 1 does not support authentication. If you are sending and receiving RIP Version 2 packets, you can enable RIP authentication on an interface. The key chain specifies the set of keys that can be used on the interface. If a key chain is not configured, no authentication is performed, not even the default.

The Device supports two modes of authentication on interfaces for which RIP authentication is enabled: plain text and MD5. The default is plain text.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip rip authentication key-chain** *name-of-chain* **Example:** `Device(config-if)# ip rip authentication key-chain trees`	Enables RIP authentication.
Step 5	**ip rip authentication mode** {**text** | **md5**} **Example:** `Device(config-if)# ip rip authentication mode md5`	Configures the interface to use plain text authentication (the default) or MD5 digest authentication.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Configuring Summary Addresses and Split Horizon

Note In general, disabling split horizon is not recommended unless you are certain that your application requires it to properly advertise routes.

If you want to configure an interface running RIP to advertise a summarized local IP address pool on a network access server for dial-up clients, use the **ip summary-address rip** interface configuration command.

Note If split horizon is enabled, neither autosummary nor interface IP summary addresses are advertised.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)# ip address 10.1.1.10 255.255.255.0`	Configures the IP address and IP subnet.
Step 5	**ip summary-address rip ip address** *ip-network mask* **Example:**	Configures the IP address to be summarized and the IP network mask.

	Command or Action	Purpose
	`Device(config-if)# ip summary-address rip ip address 10.1.1.30 255.255.255.0`	
Step 6	**no ip split horizon** **Example:** `Device(config-if)# no ip split horizon`	Disables split horizon on the interface.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show ip interface** *interface-id* **Example:** `Device# show ip interface gigabitethernet 1/0/1`	Verifies your entries.
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Split Horizon

Routers connected to broadcast-type IP networks and using distance-vector routing protocols normally use the split-horizon mechanism to reduce the possibility of routing loops. Split horizon blocks information about routes from being advertised by a router on any interface from which that information originated. This feature can optimize communication among multiple routers, especially when links are broken.

Note In general, we do not recommend disabling split horizon unless you are certain that your application requires it to properly advertise routes.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)# ip address 10.1.1.10 255.255.255.0`	Configures the IP address and IP subnet.
Step 5	**no ip split-horizon** **Example:** `Device(config-if)# no ip split-horizon`	Disables split horizon on the interface.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show ip interface** *interface-id* **Example:** `Device# show ip interface gigabitethernet 1/0/1`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuration Example for Summary Addresses and Split Horizon

In this example, the major net is 10.0.0.0. The summary address 10.2.0.0 overrides the autosummary address of 10.0.0.0 so that 10.2.0.0 is advertised out interface Gigabit Ethernet port 2, and 10.0.0.0 is not advertised. In the example, if the interface is still in Layer 2 mode (the default), you must enter a **no switchport** interface configuration command before entering the **ip address** interface configuration command.

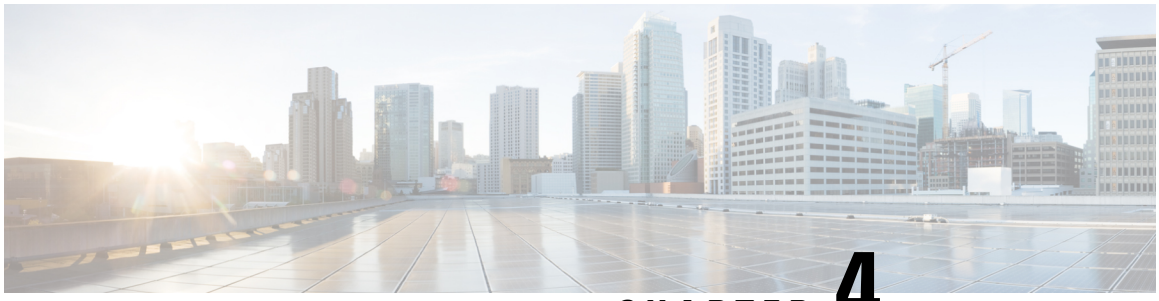
Note If split horizon is enabled, neither autosummary nor interface summary addresses (those configured with the **ip summary-address rip** router configuration command) are advertised.

```
Device(config)# router rip
Device(config-router)# interface gigabitethernet1/0/2
Device(config-if)# ip address 10.1.5.1 255.255.255.0
Device(config-if)# ip summary-address rip 10.2.0.0 255.255.0.0
Device(config-if)# no ip split-horizon
Device(config-if)# exit
Device(config)# router rip
Device(config-router)# network 10.0.0.0
Device(config-router)# neighbor 2.2.2.2 peer-group mygroup
Device(config-router)# end
```

Feature Information for Routing Information Protocol

Table 10: Feature Information for IP Unicast Routing

Feature Name	Release	Feature Information
Routing Information Protocol	Cisco IOS XE Everest 16.6.1	This feature was introduced.



CHAPTER 4

Configuring OSPF

Information About OSPF

OSPF is an Interior Gateway Protocol (IGP) designed expressly for IP networks, supporting IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending and receiving packets. The Cisco implementation supports RFC 1253, OSPF management information base (MIB).

The Cisco implementation conforms to the OSPF Version 2 specifications with these key features:

- Definition of stub areas is supported.
- Routes learned through any IP routing protocol can be redistributed into another IP routing protocol. At the intradomain level, this means that OSPF can import routes learned through EIGRP and RIP. OSPF routes can also be exported into RIP.
- Plain text and MD5 authentication among neighboring routers within an area is supported.
- Configurable routing interface parameters include interface output cost, retransmission interval, interface transmit delay, router priority, router dead and hello intervals, and authentication key.
- Virtual links are supported.
- Not-so-stubby-areas (NSSAs) per RFC 1587are supported.

OSPF typically requires coordination among many internal routers, area border routers (ABRs) connected to multiple areas, and autonomous system boundary routers (ASBRs). The minimum configuration would use all default parameter values, no authentication, and interfaces assigned to areas. If you customize your environment, you must ensure coordinated configuration of all routers.

OSPF Nonstop Forwarding

The Device or switch stack supports two levels of nonstop forwarding (NSF):

OSPF NSF Awareness

When the neighboring router is NSF-capable, the Layer 3 Device continues to forward packets from the neighboring router during the interval between the primary Route Processor (RP) in a router crashing and the backup RP taking over, or while the primary RP is manually reloaded for a non-disruptive software upgrade.

This feature cannot be disabled.

OSPF NSF Capability

supports the OSPFv2 NSF IETF format in addition to the OSPFv2 NSF Cisco format that is supported in earlier releases. For information about this feature, see : *NSF—OSPF (RFC 3623 OSPF Graceful Restart)*.

The also supports OSPF NSF-capable routing for IPv4 for better convergence and lower traffic loss following a stack's active switch change.

Note OSPF NSF requires that all neighbor networking devices be NSF-aware. If an NSF-capable router discovers non-NSF aware neighbors on a network segment, it disables NSF capabilities for that segment. Other network segments where all devices are NSF-aware or NSF-capable continue to provide NSF capabilities.

Use the **nsf** OSPF routing configuration command to enable OSPF NSF routing. Use the **show ip ospf** privileged EXEC command to verify that it is enabled.

For more information, see *Cisco Nonstop Forwarding*:
http://www.cisco.com/en/US/docs/ios/ha/configuration/guide/ha-nonstp_fwdg.html

OSPF Area Parameters

You can optionally configure several OSPF area parameters. These parameters include authentication for password-based protection against unauthorized access to an area, stub areas, and not-so-stubby-areas (NSSAs). Stub areas are areas into which information on external routes is not sent. Instead, the area border router (ABR) generates a default external route into the stub area for destinations outside the autonomous system (AS). An NSSA does not flood all LSAs from the core into the area, but can import AS external routes within the area by redistribution.

Route summarization is the consolidation of advertised addresses into a single summary route to be advertised by other areas. If network numbers are contiguous, you can use the **area range** router configuration command to configure the ABR to advertise a summary route that covers all networks in the range.

Other OSPF Parameters

You can optionally configure other OSPF parameters in router configuration mode.

- Route summarization: When redistributing routes from other protocols. Each route is advertised individually in an external LSA. To help decrease the size of the OSPF link state database, you can use the **summary-address** router configuration command to advertise a single router for all the redistributed routes included in a specified network address and mask.

- Virtual links: In OSPF, all areas must be connected to a backbone area. You can establish a virtual link in case of a backbone-continuity break by configuring two Area Border Routers as endpoints of a virtual link. Configuration information includes the identity of the other virtual endpoint (the other ABR) and the nonbackbone link that the two routers have in common (the transit area). Virtual links cannot be configured through a stub area.
- Default route: When you specifically configure redistribution of routes into an OSPF routing domain, the route automatically becomes an autonomous system boundary router (ASBR). You can force the ASBR to generate a default route into the OSPF routing domain.
- Domain Name Server (DNS) names for use in all OSPF **show** privileged EXEC command displays makes it easier to identify a router than displaying it by router ID or neighbor ID.
- Default Metrics: OSPF calculates the OSPF metric for an interface according to the bandwidth of the interface. The metric is calculated as *ref-bw* divided by bandwidth, where *ref* is 10 by default, and bandwidth (*bw*) is specified by the **bandwidth** interface configuration command. For multiple links with high bandwidth, you can specify a larger number to differentiate the cost on those links.
- Administrative distance is a rating of the trustworthiness of a routing information source, an integer between 0 and 255, with a higher value meaning a lower trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored. OSPF uses three different administrative distances: routes within an area (interarea), routes to another area (interarea), and routes from another routing domain learned through redistribution (external). You can change any of the distance values.
- Passive interfaces: Because interfaces between two devices on an Ethernet represent only one network segment, to prevent OSPF from sending hello packets for the sending interface, you must configure the sending device to be a passive interface. Both devices can identify each other through the hello packet for the receiving interface.
- Route calculation timers: You can configure the delay time between when OSPF receives a topology change and when it starts the shortest path first (SPF) calculation and the hold time between two SPF calculations.
- Log neighbor changes: You can configure the router to send a syslog message when an OSPF neighbor state changes, providing a high-level view of changes in the router.

LSA Group Pacing

The OSPF LSA group pacing feature allows the router to group OSPF LSAs and pace the refreshing, check-summing, and aging functions for more efficient router use. This feature is enabled by default with a 4-minute default pacing interval, and you will not usually need to modify this parameter. The optimum group pacing interval is inversely proportional to the number of LSAs the router is refreshing, check-summing, and aging. For example, if you have approximately 10,000 LSAs in the database, decreasing the pacing interval would benefit you. If you have a very small database (40 to 100 LSAs), increasing the pacing interval to 10 to 20 minutes might benefit you slightly.

Loopback Interfaces

OSPF uses the highest IP address configured on the interfaces as its router ID. If this interface is down or removed, the OSPF process must recalculate a new router ID and resend all its routing information out its interfaces. If a loopback interface is configured with an IP address, OSPF uses this IP address as its router

ID, even if other interfaces have higher IP addresses. Because loopback interfaces never fail, this provides greater stability. OSPF automatically prefers a loopback interface over other interfaces, and it chooses the highest IP address among all loopback interfaces.

How to Configure OSPF

Default OSPF Configuration

Table 11: Default OSPF Configuration

Feature	Default Setting
Interface parameters	Cost: Retransmit interval: 5 seconds. Transmit delay: 1 second. Priority: 1. Hello interval: 10 seconds. Dead interval: 4 times the hello interval. No authentication. No password specified. MD5 authentication disabled.
Area	Authentication type: 0 (no authentication). Default cost: 1. Range: Disabled. Stub: No stub area defined. NSSA: No NSSA area defined.
Auto cost	100 Mb/s.
Default-information originate	Disabled. When enabled, the default metric setting is 10, and the external route ty is Type 2.
Default metric	Built-in, automatic metric translation, as appropriate for each routing protocol.
Distance OSPF	dist1 (all routes within an area): 110. dist2 (all routes from one area to another): dist3 (routes from other routing domains): 110.
OSPF database filter	Disabled. All outgoing link-state advertisements (LSAs) are flooded to the interf
IP OSPF name lookup	Disabled.
Log adjacency changes	Enabled.
Neighbor	None specified.

Feature	Default Setting
Neighbor database filter	Disabled. All outgoing LSAs are flooded to the neighbor.
Network area	Disabled.
Router ID	No OSPF routing process defined.
Summary address	Disabled.
Timers LSA group pacing	240 seconds.
Timers shortest path first (spf)	spf delay: 50 miliseconds; spf-holdtime: 200 miliseconds.
Virtual link	No area ID or router ID defined. Hello interval: 10 seconds. Retransmit interval: 5 seconds. Transmit delay: 1 second. Dead interval: 40 seconds. Authentication key: no key predefined. Message-digest key (MD5): no key predefined.

Configuring Basic OSPF Parameters

To enable OSPF, create an OSPF routing process, specify the range of IP addresses to associate with the routing process, and assign area IDs to be associated with that range.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router ospf** process-id **Example:** `Device(config)# router ospf 15`	Enables OSPF routing, and enter router configuration mode. The process ID is an internally used identification parameter that is locally assigned and can be any positive integer. Each OSPF routing process has a unique value. **Note** OSPF for Routed Access supports only one OSPFv2 and one OSPFv3 instance with a maximum number of 1000 dynamically learned routes.

	Command or Action	Purpose
Step 3	**network** address wildcard-mask **area** area-id **Example:** `Device(config)# network 10.1.1.1 255.240.0.0 area 20`	Define an interface on which OSPF runs and the area ID for that interface. You can use the wildcard-mask to use a single command to define one or more multiple interfaces to be associated with a specific OSPF area. The area ID can be a decimal value or an IP address.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show ip protocols** **Example:** `Device# show ip protocols`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring OSPF Interfaces

You can use the **ip ospf** interface configuration commands to modify interface-specific OSPF parameters. You are not required to modify any of these parameters, but some interface parameters (hello interval, dead interval, and authentication key) must be consistent across all routers in an attached network. If you modify these parameters, be sure all routers in the network have compatible values.

Note The **ip ospf** interface configuration commands are all optional.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**interface** interface-id **Example:**	Enters interface configuration mode, and specifies the Layer 3 interface to configure.

	Command or Action	Purpose
	`Device(config)# interface gigabitethernet 1/0/1`	
Step 3	**ip ospf** cost **Example:** `Device(config-if)# ip ospf 8`	(Optional) Explicitly specifies the cost of sending a packet on the interface.
Step 4	**ip ospf retransmit-interval** seconds **Example:** `Device(config-if)# ip ospf transmit-interval 10`	(Optional) Specifies the number of seconds between link state advertisement transmissions. The range is 1 to 65535 seconds. The default is 5 seconds.
Step 5	**ip ospf transmit-delay** seconds **Example:** `Device(config-if)# ip ospf transmit-delay 2`	(Optional) Sets the estimated number of seconds to wait before sending a link state update packet. The range is 1 to 65535 seconds. The default is 1 second.
Step 6	**ip ospf priority** number **Example:** `Device(config-if)# ip ospf priority 5`	(Optional) Sets priority to help find the OSPF designated router for a network. The range is from 0 to 255. The default is 1.
Step 7	**ip ospf hello-interval** seconds **Example:** `Device(config-if)# ip ospf hello-interval 12`	(Optional) Sets the number of seconds between hello packets sent on an OSPF interface. The value must be the same for all nodes on a network. The range is 1 to 65535 seconds. The default is 10 seconds.
Step 8	**ip ospf dead-interval** seconds **Example:** `Device(config-if)# ip ospf dead-interval 8`	(Optional) Sets the number of seconds after the last device hello packet was seen before its neighbors declare the OSPF router to be down. The value must be the same for all nodes on a network. The range is 1 to 65535 seconds. The default is 4 times the hello interval.
Step 9	**ip ospf authentication-key** key **Example:** `Device(config-if)# ip ospf authentication-key password`	(Optional) Assign a password to be used by neighboring OSPF routers. The password can be any string of keyboard-entered characters up to 8 bytes in length. All neighboring routers on the same network must have the same password to exchange OSPF information.
Step 10	**ip ospf message digest-key** keyid **md5** key **Example:** `Device(config-if)# ip ospf message digest-key 16 md5 your1pass`	(Optional) Enables MDS authentication. • *keyid*—An identifier from 1 to 255. • *key*—An alphanumeric password of up to 16 bytes.
Step 11	**ip ospf database-filter all out** **Example:**	(Optional) Block flooding of OSPF LSA packets to the interface. By default, OSPF floods new LSAs over all

	Command or Action	Purpose
	`Device(config-if)# ip ospf database-filter all out`	interfaces in the same area, except the interface on which the LSA arrives.
Step 12	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 13	**show ip ospf interface** [*interface-name*] **Example:** `Device# show ip ospf interface`	Displays OSPF-related interface information.
Step 14	**show ip ospf neighbor detail** **Example:** `Device# show ip ospf neighbor detail`	Displays NSF awareness status of neighbor switch. The output matches one of these examples: • *Options is 0x52* *LLS Options is 0x1 (LR)* When both of these lines appear, the neighbor switch is NSF aware. • *Options is 0x42*—This means the neighbor switch is not NSF aware.
Step 15	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring OSPF Area Parameters

Before you begin

Note The OSPF **area** router configuration commands are all optional.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters the global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 2	**router ospf** process-id **Example:** `Device(config)# router ospf 109`	Enables OSPF routing, and enter router configuration mode.
Step 3	**area** *area-id* **authentication** **Example:** `Device(config-router)# area 1 authentication`	(Optional) Allow password-based protection against unauthorized access to the identified area. The identifier can be either a decimal value or an IP address.
Step 4	**area** *area-id* **authentication message-digest** **Example:** `Device(config-router)# area 1 authentication message-digest`	(Optional) Enables MD5 authentication on the area.
Step 5	**area** *area-id* **stub** [**no-summary**] **Example:** `Device(config-router)# area 1 stub`	(Optional) Define an area as a stub area. The **no-summary** keyword prevents an ABR from sending summary link advertisements into the stub area.
Step 6	**area** *area-id* **nssa** [**no-redistribution**] [**default-information-originate**] [**no-summary**] **Example:** `Device(config-router)# area 1 nssa default-information-originate`	(Optional) Defines an area as a not-so-stubby-area. Every router within the same area must agree that the area is NSSA. Select one of these keywords: • **no-redistribution**—Select when the router is an NSSA ABR and you want the **redistribute** command to import routes into normal areas, but not into the NSSA. • **default-information-originate**—Select on an ABR to allow importing type 7 LSAs into the NSSA. • **no-redistribution**—Select to not send summary LSAs into the NSSA.
Step 7	**area** *area-id* **range** *address mask* **Example:** `Device(config-router)# area 1 range 255.240.0.0`	(Optional) Specifies an address range for which a single route is advertised. Use this command only with area border routers.
Step 8	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 9	**show ip ospf** [*process-id*] **Example:** `Device# show ip ospf`	Displays information about the OSPF routing process in general or for a specific process ID to verify configuration.
Step 10	**show ip ospf** [*process-id* [*area-id*]] **database** **Example:** `Device# show ip osfp database`	Displays lists of information related to the OSPF database for a specific router.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Other OSPF Parameters

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router ospf** *process-id* **Example:** `Device(config)# router ospf 10`	Enables OSPF routing, and enter router configuration mode.
Step 3	**summary-address** *address mask* **Example:** `Device(config)# summary-address 10.1.1.1 255.255.255.0`	(Optional) Specifies an address and IP subnet mask for redistributed routes so that only one summary route is advertised.
Step 4	**area** *area-id* **virtual-link** *router-id* [**hello-interval** *seconds*] [**retransmit-interval** *seconds*] [**trans**] [[**authentication-key** *key*] | **message-digest-key** *keyid* **md5** *key*]] **Example:** `Device(config)# area 2 virtual-link 192.168.255.1 hello-interval 5`	(Optional) Establishes a virtual link and set its parameters.

	Command or Action	Purpose
Step 5	**default-information originate** [**always**] [**metric** *metric-value*] [**metric-type** *type-value*] [**route-map** *map-name*] **Example:** `Device(config)# default-information originate metric 100 metric-type 1`	(Optional) Forces the ASBR to generate a default route into the OSPF routing domain. Parameters are all optional.
Step 6	**ip ospf name-lookup** **Example:** `Device(config)# ip ospf name-lookup`	(Optional) Configures DNS name lookup. The default is disabled.
Step 7	**ip auto-cost reference-bandwidth** *ref-bw* **Example:** `Device(config)# ip auto-cost reference-bandwidth 5`	(Optional) Specifies an address range for which a single route will be advertised. Use this command only with area border routers.
Step 8	**distance ospf** {[**inter-area** *dist1*] [**inter-area** *dist2*] [**external** *dist3*]} **Example:** `Device(config)# distance ospf inter-area 150`	(Optional) Changes the OSPF distance values. The default distance for each type of route is 110. The range is 1 to 255.
Step 9	**passive-interface** *type number* **Example:** `Device(config)# passive-interface gigabitethernet 1/0/6`	(Optional) Suppresses the sending of hello packets through the specified interface.
Step 10	**timers throttle spf** *spf-delay spf-holdtime spf-wait* **Example:** `Device(config)# timers throttle spf 200 100 100`	(Optional) Configures route calculation timers. • *spf-delay*—Delay between receiving a change to SPF calculation. The range is from 1 to 600000 in miliseconds. • *spf-holdtime*—Delay between first and second SPF calculation. The range is from 1 to 600000 in milliseconds. • *spf-wait*—Maximum wait time in milliseconds for SPF calculations. The range is from 1 to 600000 in milliseconds.
Step 11	**ospf log-adj-changes** **Example:** `Device(config)# ospf log-adj-changes`	(Optional) Sends syslog message when a neighbor state changes.

	Command or Action	Purpose
Step 12	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 13	**show ip ospf** [*process-id* [*area-id*]] **database** **Example:** `Device# show ip ospf database`	Displays lists of information related to the OSPF database for a specific router.
Step 14	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Changing LSA Group Pacing

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router ospf** *process-id* **Example:** `Device(config)# router ospf 25`	Enables OSPF routing, and enter router configuration mode.
Step 3	**timers lsa-group-pacing** *seconds* **Example:** `Device(config-router)# timers lsa-group-pacing 15`	Changes the group pacing of LSAs.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:**	Verifies your entries.

	Command or Action	Purpose
	`Device# show running-config`	
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring a Loopback Interface

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**interface loopback 0** **Example:** `Device(config)# interface loopback 0`	Creates a loopback interface, and enter interface configuration mode.
Step 3	**ip address** address mask **Example:** `Device(config-if)# ip address 10.1.1.5 255.255.240.0`	Assign an IP address to this interface.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show ip interface** **Example:** `Device# show ip interface`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Monitoring OSPF

You can display specific statistics such as the contents of IP routing tables, caches, and databases.

Table 12: Show IP OSPF Statistics Commands

show ip ospf [*process-id*]	Displays general inform processes.
show ip ospf [*process-id*] **database** [**router**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**router**] [**self-originate**] **show ip ospf** [*process-id*] **database** [**router**] [**adv-router** [*ip-address*]] **show ip ospf** [*process-id*] **database** [**network**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**summary**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**asbr-summary**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**external**] [*link-state-id*] **show ip ospf** [*process-id area-id*] **database** [**database-summary**]	Displays lists of informa
show ip ospf border-routes	Displays the internal OS entries.
show ip ospf interface [*interface-name*]	Displays OSPF-related
show ip ospf neighbor [*interface-name*] [*neighbor-id*] **detail**	Displays OSPF interfac
show ip ospf virtual-links	Displays OSPF-related

Configuration Examples for OSPF

Example: Configuring Basic OSPF Parameters

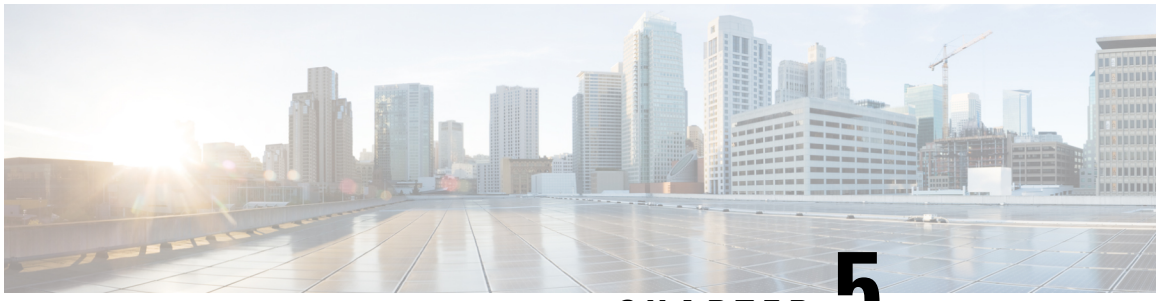
This example shows how to configure an OSPF routing process and assign it a process number of 109:

```
Device(config)# router ospf 109
Device(config-router)# network 131.108.0.0 255.255.255.0 area 24
```

Feature Information for OSPF

Table 13: Feature Information for OSPF

Feature Name	Release	Feature Information
OSPF (Open Shortest Path First)	Cisco IOS XE Everest 16.6.1	This feature was introduced



CHAPTER 5

Configuring OSPFv3 Authentication Support with IPsec

Information About OSPFv3 Authentication Support with IPsec

The following sections provide information about OSPFv3 authentication support with IPsec and OSPFv3 virtual links.

Overview of OSPFv3 Authentication Support with IPsec

In order to ensure that OSPFv3 packets are not altered and resent to the device, causing the device to behave in a way not desired by its system administrators, OSPFv3 packets must be authenticated. OSPFv3 uses the IPsec secure socket to add authentication to OSPFv3 packets.

OSPFv3 requires the use of IPsec to enable authentication. Crypto images are required to use authentication because only crypto images include the IPsec needed for use with OSPFv3.

In OSPFv3, authentication fields have been removed from OSPFv3 packet headers. When OSPFv3 runs on IPv6, OSPFv3 requires the IPv6 authentication header or IPv6 Encapsulating Security Payload (ESP) header to ensure integrity, authentication, and confidentiality of routing exchanges. IPv6 authentication header and ESP extension headers can be used to provide authentication and confidentiality to OSPFv3.

To use the IPsec authentication header, you must enable the **ipv6 ospf authentication** command. To use the IPsec ESP header, you must enable the **ipv6 ospf encryption** command. The ESP header can be applied alone or along with the authentication header, and when ESP is used, both encryption and authentication are provided. Security services can be provided between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host.

To configure IPsec, you should configure a security policy, which is a combination of the security policy index (SPI) and the key (the key is used to create and validate the hash value). IPsec for OSPFv3 can be configured on an interface or on an OSPFv3 area. For higher security, you should configure a different policy

on each interface that is configured with IPsec. If you configure IPsec for an OSPFv3 area, the policy is applied to all the interfaces in that area, except for the interfaces that have IPsec configured directly. After IPsec is configured for OSPFv3, IPsec is invisible to you.

The IPsecure socket is used by applications to secure traffic by allowing the application to open, listen, and close secure sockets. The binding between the application and the secure socket layer also allows the secure socket layer to inform the application of changes to the socket, such as connection open and close events. The IPsecure socket is able to identify the socket, that is, it can identify the local and remote addresses, masks, ports, and protocol that carry the traffic requiring security.

Each interface has a secure socket state, which can be one of the following:

- NULL: Do not create a secure socket for the interface if authentication is configured for the area.
- DOWN: IPsec has been configured for the interface (or the area that contains the interface), but OSPFv3 has either not requested IPsec to create a secure socket for this interface, or there is an error condition.

Note OSPFv3 does not send or accept packets while in the DOWN state.

- GOING UP: OSPFv3 has requested a secure socket from IPsec and is waiting for a CRYPTO_SS_SOCKET_UP message from IPsec.
- UP: OSPFv3 has received a CRYPTO_SS_SOCKET_UP message from IPsec.
- CLOSING: The secure socket for the interface has been closed. A new socket can be opened for the interface, in which case, the current secure socket makes the transition to the DOWN state. Otherwise, the interface becomes UNCONFIGURED.
- UNCONFIGURED: Authentication is not configured on the interface.

OSPFv3 Virtual Links

For each virtual link, a primary security information data block is created. Because a secure socket must be opened on each interface, there will be a corresponding security information datablock for each interface in the transit area. The secure socket state is kept in the interface's security information datablock. The **state** field in the primary security information datablock shows the status of all the secure sockets opened for the corresponding virtual link. If all the secure sockets are UP, the security state for the virtual link is set to UP.

Packets sent on a virtual link with IPsec must use predetermined source and destination addresses. The first local area address found in the device's intra-area-prefix Link-State Advertisement (LSA) for the area is used as the source address. This source address is saved in the area's data structure and used when secure sockets are opened and packets sent over the corresponding virtual link. The virtual link does not transition to the point-to-point state until a source address is selected. Also, when the source or destination address changes, the previous secure sockets must be closed and new secure sockets opened.

Note Virtual links are not supported for the IPv4 address family.

How to Configure OSPFv3 Authentication Support with IPsec

The following sections provide information on how to define authentication on an interface, and how to define authentication in an OSPFv3 area.

Defining Authentication on an Interface

To define authentication on an interface, perform this procedure:

Before you begin

Before you configure IPsec on an interface, you must configure OSPFv3 on that interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 1/0/1`	Configures an interface.

Step 4	Choose one of the following:
	• **ospfv3 authentication** {{**ipsec spi** *spi* {**md5** | **sha1**}} | { *key-encryption-type key* } | **null**}
	• **ipv6 ospf authentication** {**null** | **ipsec spi** *spi* *authentication-algorithm* [*key-encryption-type*] [*key*]}
	Example:
	`Device(config-if)# ospfv3 authentication md5 0 275761340947681324733020312097 27`
	OR
	`Device(config-if)# ipv6 ospf authentication ipsec spi 500 md5 1234567890abcdef1234567890abcdef`
		Specifies the authentication type for an interface.

Defining Authentication in an OSPFv3 Area

To define authentication in an OSPFv3 area, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 1`	Enables OSPFv3 router configuration mode.
Step 4	**area** *area-id* **authentication ipsec spi** *spi authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 authentication ipsec spi 678 md5 1234567890ABCDEF1234567890ABCDEF`	Enables authentication in an OSPFv3 area.

How to Configure OSPFv3 IPSec ESP Encryption and Authentication

The following sections provide information on how to define encryption on an interface, how to define encryption in an OSPFv3 area, and how to defining authentication and encryption for a virtual link in an OSPFv3 area:

Defining Encryption on an Interface

To define encryption on an interface, perform this procedure.

Before you begin

Before you configure IPsec on an interface, you must configure OSPFv3 on that interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 1/0/1`	Configures an interface.

Step 4	Choose one of the following:
	• **ospfv3 authentication** {**ipsec spi** *spi* **esp** *encryption-algorithm key-encryption-type key authentication-algorithm key-encryption-type key* | **null**}
	• **ipv6 ospf authentication** {**ipsec spi** *spi* **esp** {*encryption-algorithm* [*key-encryption-type*] *key* | **null**} *authentication-algorithm* [*key-encryption-type*] *key*] | **null**}
	Example:
	`Device(config-if)# ospfv3 encryption ipsec spi 1001 esp null md5 0 275761340947681324733020312097 27`
	OR
	`Device(config-if)# ipv6 ospf encryption ipsec spi 1001 esp null sha1 123456789A123456789B123456789C123456789D`
		Specifies the encryption type for the interface.

Defining Encryption in an OSPFv3 Area

To define encryption in an OSPFv3 area, perform this procedure.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 1`	Enables OSPFv3 router configuration mode.

	Command or Action	Purpose
Step 4	**area** *area-id* **encryption ipsec spi** *spi* **esp** {*encryption-algorithm* [*key-encryption-type*] *key* | **null**} *authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 encryption ipsec spi 500 esp null md5 1aaa2bbb3ccc4ddd5eee6fff7aaa8bbb`	Enables encryption in an OSPFv3 area.

Defining Authentication and Encryption for a Virtual Link in an OSPFv3 Area

To define authentication and encryption for a virtual link in an OSPFv3 area, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 1`	Enables OSPFv3 router configuration mode.
Step 4	**area** *area-id* **virtual-link** *router-id* **authentication ipsec spi** *spi authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 virtual-link 10.0.0.1 authentication ipsec spi 940 md5 1234567890ABCDEF1234567890ABCDEF`	Enables authentication for virtual links in an OSPFv3 area.
Step 5	**area** *area-id* **virtual-link** *router-id* **authentication ipsec spi** *spi* **esp** {*encryption-algorithm* [*key-encryption-type*] *key* | **null**} *authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 virtual-link 10.1.0.1 hello-interval 2 dead-interval 10 encryption ipsec spi 3944 esp null sha1 123456789A123456789B123456789C123456789D`	Enables encryption for virtual links in the OSPFv3 area.

Configuration Examples for OSPFv3 Authentication Support with IPsec

The following sections provide various configuration examples for OSPFv3 authentication support with IPsec.

Example: Defining Authentication on an Interface

The following example shows how to define authentication on Ethernet interface 1/0/1:

```
Device> enable
Device# configure terminal
Device(config)# interface Ethernet1/0/1
Device(config-if)# ipv6 enable
Device(config-if)# ipv6 ospf 1 area 0
Device(config-if)# ipv6 ospf authentication ipsec spi 500 md5 1234567890ABCDEF1234567890ABCDEF
Device(config-if)# exit
Device(config)# interface Ethernet1/0/1
Device(config-if)# ipv6 enable
Device(config-if)# ipv6 ospf authentication null
Device(config-if)# ipv6 ospf 1 area 0
```

Example: Defining Authentication in an OSPFv3 Area

The following example shows how to define authentication on OSPFv3 area 0:

```
Device> enable
Device# configure terminal
Device(config)# ipv6 router ospf 1
Device(config-router)# router-id 10.11.11.1
Device(config-router)# area 0 authentication ipsec spi 1000 md5
1234567890ABCDEF1234567890ABCDEF
```

Configuration Example for OSPFv3 IPSec ESP Encryption and Authentication

The following section provides an example to verify OSPFv3 IPsec ESP encryption and authentication.

Example: Verifying Encryption in an OSPFv3 Area

The following is a sample output of the **show ipv6 ospf interface** command:

```
Device> enable
Device# show ipv6 ospf interface
```

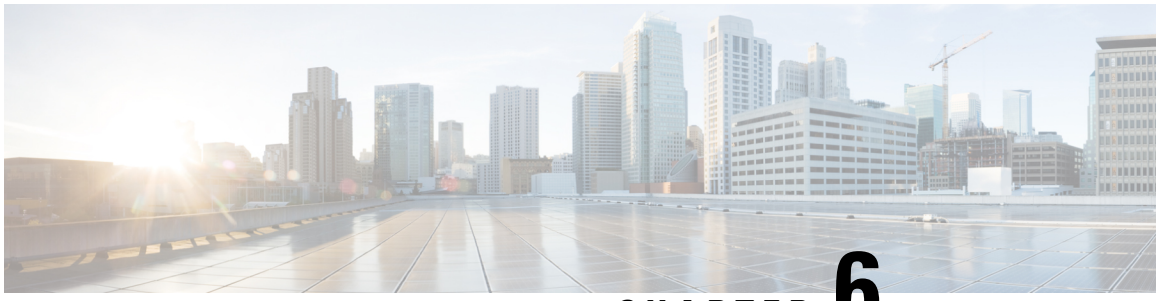
```
Ethernet1/0/1 is up, line protocol is up
  Link Local Address 2001:0DB1:A8BB:CCFF:FE00:6E00, Interface ID 2
  Area 0, Process ID 1, Instance ID 0, Router ID 10.10.10.1
  Network Type BROADCAST, Cost:10
  MD5 Authentication (Area) SPI 1000, secure socket state UP (errors:0)
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 10.11.11.1, local address 2001:0DB1:A8BB:CCFF:FE00:6F00
  Backup Designated router (ID) 10.10.10.1, local address
FE80::A8BB:CCFF:FE00:6E00
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:03
  Index 1/1/1, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.11.11.1  (Designated Router)
  Suppress hello for 0 neighbor(s)
```

Feature History and Information for OSPFv3 Authentication Support with IPsec

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 14: Feature History for OSPFv3 Authentication Support with IPsec

Feature Name	Release	Feature Information
OSPFv3 Authentication Support with IPsec	Cisco IOS XE Fuji 16.8.1a	OSPFv3 uses the IPsec secure socket to add authentication to OSPFv3 packets.



CHAPTER 6

Configuring OSPFv3 Authentication Trailer

Information About the OSPFv3 Authentication Trailer

The OSPFv3 authentication trailer feature (as defined in RFC 7166) provides an alternative mechanism to authenticate Open Shortest Path First version 3 (OSPFv3) protocol packets. Prior to the OSPFv3 authentication trailer, OSPFv3 IPsec (as defined in RFC 4552) was the only mechanism for authenticating protocol packets. The OSPFv3 authentication trailer feature also provides packet replay protection through sequence number and do not have platform dependencies.

To perform non-IPsec cryptographic authentication, devices attach a special data block, that is, authentication trailer, to the end of the OSPFv3 packet. The length of the authentication trailer is not included in the length of the OSPFv3 packet but is included in the IPv6 payload length. The Link-Local Signaling (LLS) block is established by the L-bit setting in the **OSPFv3 Options** field in OSPFv3 hello packets and database description packets. If present, the LLS data block is included in the cryptographic authentication computation along with the OSPFv3 packet.

A new authentication trailer bit is introduced into the **OSPFv3 Options** field. OSPFv3 devices must set the authentication trailer bit in OSPFv3 hello packets and database description packets to indicate that all the packets on this link will include an authentication trailer. For OSPFv3 hello packets and database description packets, the authentication trailer bit indicates the authentication trailer is present. For other OSPFv3 packet types, the OSPFv3 authentication trailer bit setting from the OSPFv3 hello and database description setting is preserved in the OSPFv3 neighbor data structure. OSPFv3 packet types that do not include the **OSPFv3 Options** field uses the setting from the neighbor data structure to determine whether or not the authentication trailer is expected. The authentication trailer bit must be set in all OSPFv3 hello packets and database description packets that contain an authentication trailer.

To configure the authentication trailer, OSPFv3 utilizes the existing Cisco IOS **key chain** command. For outgoing OSPFv3 packets, the following rules are used to select the key from the key chain:

- Select the key that is the last to expire.
- If two keys have the same stop time, select the one with the highest key ID.

The security association ID maps to the authentication algorithm and the secret key that is used to generate and verify the message digest. If the authentication is configured, but the last valid key is expired, the packets are sent using the key. A syslog message is also generated. If no valid key is available, the packet is sent without the authentication trailer. When packets are received, the key ID is used to look up the data for that key. If the key ID is not found in the key chain, or if the security association is not valid, the packet is dropped. Otherwise, the packet is verified using the algorithm and the key that is configured for the key ID. Key chains support rollover using key lifetimes. A new key can be added to a key chain with the send start time set in the future. This setting allows the new key to be configured on all the devices before the keys are actually used.

The hello packets have higher priority than other OSPFv3 packets, and therefore, can get reordered on the outgoing interface. This reordering can create problems with sequence number verification on neighboring devices. To prevent sequence mismatch, OSPFv3 verifies the sequence number separately for each packet type. See RFC 7166 for more details on the authentication procedure.

During the initial rollover of the authentication trailer feature on the network, adjacency can be maintained between the devices configured with authentication routes and devices that are yet to be configured by using the deployment mode. When the deployment mode is configured using the **authentication mode deployment** command, the packets are processed differently. For the outgoing packets, OSPF checksum is calculated even if authentication trailer is configured. For incoming packets, the packets without authentication trailer or the wrong authentication hash are dropped. In the deployment mode, the **show ospfv3 neighbor** *detail* command shows the last packet authentication status. This information can be used to verify if the authentication trailer feature is working before the mode is set to normal with the **authentication mode normal** command.

How to Configure the OSPFv3 Authentication Trailer

To configure OSPFv3 authentication trailer, perform this procedure:

Before you begin

An authentication key is required for configuring OSPFv3 authentication trailer. For more information on configuring an authentication key, see *How to Configure Authentication Keys* in *Protocol-Independent Features*.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 2/0/1`	Specifies the interface type and number.

	Command or Action	Purpose
Step 4	**ospfv3** [*pid*] [**ipv4** | **ipv6**] **authentication** {**key-chain** *chain-name* | **null**} **Example:** `Device(config-if)# ospfv3 1 ipv6 authentication key-chain ospf-1`	Specifies the authentication type for an OSPFv3 instance.
Step 5	**router ospfv3** [*process-id*] **Example:** `Device(config-if)# router ospfv3 1`	Enters OSPFv3 router configuration mode.
Step 6	**address-family ipv6 unicast** **Example:** `Device(config-router)# address-family ipv6 unicast`	Configures the IPv6 address family in the OSPFv3 process and enters IPv6 address family configuration mode.
Step 7	**area** *area-id* **authentication** {**key-chain** *chain-name* | **null**} **Example:** `Device(config-router-af)# area 1 authentication key-chain ospf-chain-1`	Configures the authentication trailer on all interfaces in the OSPFv3 area.
Step 8	**area** *area-id* **virtual-link** *router-id* **authentication key-chain** *chain-name* **Example:** `Device(config-router-af)# area 1 virtual-link 1.1.1.1 authentication key-chain ospf-chain-1`	Configures the authentication for virtual links.
Step 9	**area** *area-id* **sham-link** *source-address destination-address* **authentication key-chain** *chain-name* **Example:** `Device(config-router-af)# area 1 sham-link 1.1.1.1 1.1.1.0 authentication key-chain ospf-chain-1`	Configures the authentication for sham-links.
Step 10	**authentication mode** {**deployment** | **normal**} **Example:** `Device(config-router-af)# authentication mode deployment`	(Optional) Specifies the type of authentication used for the OSPFv3 instance. The **deployment** keyword provides adjacency between configured and the unconfigured authentication devices.
Step 11	**end** **Example:** `Device(config-router-af)# end`	Exits IPv6 address family configuration mode and returns to privileged EXEC mode.
Step 12	**show ospfv3 interface** **Example:** `Device# show ospfv3`	(Optional) Displays OSPFv3-related interface information.

	Command or Action	Purpose
Step 13	**show ospfv3 neighbor** [*detail*] **Example:** `Device# show ospfv3 neighbor detail`	(Optional) Displays OSPFv3 neighbor information on a per-interface basis.
Step 14	**debug ospfv3** **Example:** `Device# debug ospfv3`	(Optional) Displays debugging information for OSPFv3.

Configuration Examples for the OSPFv3 Authentication Trailer

The following sections provide examples on how to configure the OSPFv3 authentication trailer and how to verify the OSPFv3 authentication trailer configuration.

Example: Configuring the OSPFv3 Authentication Trailer

The following example shows how to define authentication trailer on GigabitEthernet interface 1/0/1:

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# ospfv3 1 ipv6 authentication key-chain ospf-1
Device(config-if)# router ospfv3 1
Device(config-router)# address-family ipv6 unicast
Device(config-router-af)# area 1 authentication key-chain ospf-1
Device(config-router-af)# area 1 virtual-link 1.1.1.1 authentication key-chain ospf-1
Device(config-router-af)# area 1 sham-link 1.1.1.1 authentication key-chain ospf-1
Device(config-router-af)# authentication mode deployment
Device(config-router-af)# end
Device(config)# key chain ospf-1
Device(config-keychain)# key 1
Device(config-keychain-key)# key-string ospf
Device(config-keychain-key)# cryptographic-algorithm hmac-sha-256
!
```

Example: Verifying OSPFv3 Authentication Trailer

The following example shows the output of the **show ospfv3** command.

```
Device# show ospfv3
 OSPFv3 1 address-family ipv6
 Router ID 1.1.1.1
…
RFC1583 compatibility enabled
 Authentication configured with deployment key lifetime
 Active Key-chains:
```

```
  Key chain ospf-1: Send key 1, Algorithm HMAC-SHA-256, Number of interfaces 1
    Area BACKBONE(0)
```

The following example shows the output of the **show ospfv3 neighbor detail** command.

```
Device# show ospfv3 neighbor detail
OSPFv3 1 address-family ipv6 (router-id 2.2.2.2)
 Neighbor 1.1.1.1
    In the area 0 via interface GigabitEthernet0/0
    Neighbor: interface-id 2, link-local address FE80::A8BB:CCFF:FE01:2D00
    Neighbor priority is 1, State is FULL, 6 state changes
    DR is 2.2.2.2 BDR is 1.1.1.1
    Options is 0x000413 in Hello (V6-Bit, E-Bit, R-Bit, AT-Bit)
    Options is 0x000413 in DBD (V6-Bit, E-Bit, R-Bit, AT-Bit)
    Dead timer due in 00:00:33
    Neighbor is up for 00:05:07
    Last packet authentication succeed
    Index 1/1/1, retransmission queue length 0, number of retransmission 0
    First 0x0(0)/0x0(0)/0x0(0) Next 0x0(0)/0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

The following example shows the output of the **show ospfv3 interface** command.

```
Device# show ospfv3 interface
GigabitEthernet1/0/1 is up, line protocol is up
  Cryptographic authentication enabled
    Sending SA: Key 25, Algorithm HMAC-SHA-256 - key chain ospf-1
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Additional References for OSPFv3 Authentication Trailer

Related Documents

Related Topic	Document Title
Configuring OSPF features	*IP Routing: OSPF Configuration Guide*

Standards and RFCs

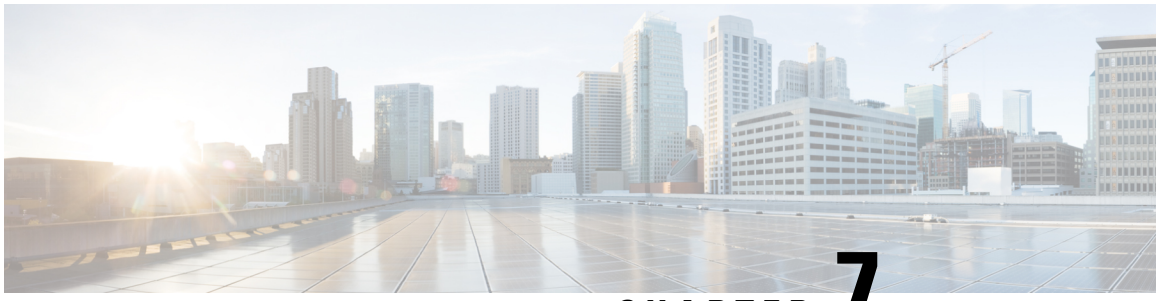
Standard/RFC	Document Title
RFC 7166	RFC for Supporting Authentication Trailer for OSPFv3
RFC 6506	RFC for Supporting Authentication Trailer for OSPFv3
RFC 4552	RFC for Authentication/Confidentiality for OSPFv3

Feature Information for the OSPFv3 Authentication Trailer

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 15: Feature Information for the OSPFv3 Authentication Trailer

Feature Name	Releases	Feature Information
OSPFv3 Authentication Trailer	Cisco IOS XE Fuji 16.8.1a	OSPFv3 Authentication Trailer feature provides a mechanism to authenticate OSPFv3 protocol packets as an alternative to existing OSPFv3 IPsec authentication.



CHAPTER 7

Configuring EIGRP

Information About EIGRP

Enhanced IGRP (EIGRP) is a Cisco proprietary enhanced version of the IGRP. EIGRP uses the same distance vector algorithm and distance information as IGRP; however, the convergence properties and the operating efficiency of EIGRP are significantly improved.

The convergence technology employs an algorithm referred to as the Diffusing Update Algorithm (DUAL), which guarantees loop-free operation at every instant throughout a route computation and allows all devices involved in a topology change to synchronize at the same time. Routers that are not affected by topology changes are not involved in recomputations.

IP EIGRP provides increased network width. With RIP, the largest possible width of your network is 15 hops. Because the EIGRP metric is large enough to support thousands of hops, the only barrier to expanding the network is the transport-layer hop counter. EIGRP increments the transport control field only when an IP packet has traversed 15 routers and the next hop to the destination was learned through EIGRP. When a RIP route is used as the next hop to the destination, the transport control field is incremented as usual.

EIGRP Features

EIGRP offers these features:

- Fast convergence.
- Incremental updates when the state of a destination changes, instead of sending the entire contents of the routing table, minimizing the bandwidth required for EIGRP packets.
- Less CPU usage because full update packets need not be processed each time they are received.
- Protocol-independent neighbor discovery mechanism to learn about neighboring routers.
- Variable-length subnet masks (VLSMs).
- Arbitrary route summarization.

- EIGRP scales to large networks.

EIGRP Components

EIGRP has these four basic components:

- Neighbor discovery and recovery is the process that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. Neighbor discovery and recovery is achieved with low overhead by periodically sending small hello packets. As long as hello packets are received, the Cisco IOS software can learn that a neighbor is alive and functioning. When this status is determined, the neighboring routers can exchange routing information.
- The reliable transport protocol is responsible for guaranteed, ordered delivery of EIGRP packets to all neighbors. It supports intermixed transmission of multicast and unicast packets. Some EIGRP packets must be sent reliably, and others need not be. For efficiency, reliability is provided only when necessary. For example, on a multiaccess network that has multicast capabilities (such as Ethernet), it is not necessary to send hellos reliably to all neighbors individually. Therefore, EIGRP sends a single multicast hello with an indication in the packet informing the receivers that the packet need not be acknowledged. Other types of packets (such as updates) require acknowledgment, which is shown in the packet. The reliable transport has a provision to send multicast packets quickly when there are unacknowledged packets pending. Doing so helps ensure that convergence time remains low in the presence of varying speed links.
- The DUAL finite state machine embodies the decision process for all route computations. It tracks all routes advertised by all neighbors. DUAL uses the distance information (known as a metric) to select efficient, loop-free paths. DUAL selects routes to be inserted into a routing table based on feasible successors. A successor is a neighboring router used for packet forwarding that has a least-cost path to a destination that is guaranteed not to be part of a routing loop. When there are no feasible successors, but there are neighbors advertising the destination, a recomputation must occur. This is the process whereby a new successor is determined. The amount of time it takes to recompute the route affects the convergence time. Recomputation is processor-intensive; it is advantageous to avoid recomputation if it is not necessary. When a topology change occurs, DUAL tests for feasible successors. If there are feasible successors, it uses any it finds to avoid unnecessary recomputation.
- The protocol-dependent modules are responsible for network layer protocol-specific tasks. An example is the IP EIGRP module, which is responsible for sending and receiving EIGRP packets that are encapsulated in IP. It is also responsible for parsing EIGRP packets and informing DUAL of the new information received. EIGRP asks DUAL to make routing decisions, but the results are stored in the IP routing table. EIGRP is also responsible for redistributing routes learned by other IP routing protocols.

Note To enable EIGRP, the Device or active switch must be running the

EIGRP Nonstop Forwarding

The Device stack supports two levels of EIGRP nonstop forwarding:

- EIGRP NSF Awareness

- EIGRP NSF Capability

EIGRP NSF Awareness

The supports EIGRP NSF Awareness for IPv4. When the neighboring router is NSF-capable, the Layer 3 Device continues to forward packets from the neighboring router during the interval between the primary Route Processor (RP) in a router failing and the backup RP taking over, or while the primary RP is manually reloaded for a nondisruptive software upgrade. This feature cannot be disabled.

EIGRP NSF Capability

The supports EIGRP Cisco NSF routing to speed up convergence and to eliminate traffic loss after a stack's active switch changeover.

The also supports EIGRP NSF-capable routing for IPv4 for better convergence and lower traffic loss following an active switch changeover. When an EIGRP NSF-capable active switch restarts or a new active switch starts up and NSF restarts, the Device has no neighbors, and the topology table is empty. The Device must bring up the interfaces, reacquire neighbors, and rebuild the topology and routing tables without interrupting the traffic directed toward the Device stack. EIGRP peer routers maintain the routes learned from the new active switch and continue forwarding traffic through the NSF restart process.

To prevent an adjacency reset by the neighbors, the new active switch uses a new Restart (RS) bit in the EIGRP packet header to show the restart. When the neighbor receives this, it synchronizes the stack in its peer list and maintains the adjacency with the stack. The neighbor then sends its topology table to the active switch with the RS bit set to show that it is NSF-aware and is aiding the new active switch.

If at least one of the stack peer neighbors is NSF-aware, the active switch receives updates and rebuilds its database. Each NSF-aware neighbor sends an end of table (EOT) marker in the last update packet to mark the end of the table content. The active switch recognizes the convergence when it receives the EOT marker, and it then begins sending updates. When the active switch has received all EOT markers from its neighbors or when the NSF converge timer expires, EIGRP notifies the routing information database (RIB) of convergence and floods its topology table to all NSF-aware peers.

EIGRP Stub Routing

The EIGRP stub routing feature reduces resource utilization by moving routed traffic closer to the end user.

Note The EIGRP stub routing capability advertises connected or summary routes from the routing tables to other device in the network. The device uses EIGRP stub routing at the access layer to eliminate the need for other types of routing advertisements. On a device running the Network Essentials license, if you try to configure multi-VRF-CE and EIGRP stub routing at the same time, the configuration is not allowed. IPv6 EIGRP stub routing is not supported with the Network Essentials license.

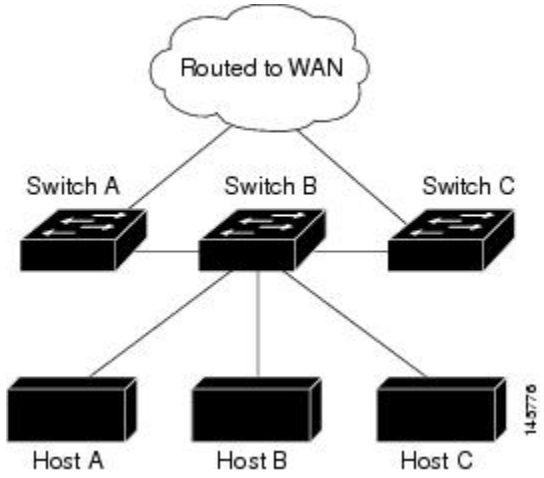
In a network using EIGRP stub routing, the only allowable route for IP traffic to the user is through a device that is configured with EIGRP stub routing. The device sends the routed traffic to interfaces that are configured as user interfaces or are connected to other devices.

When using EIGRP stub routing, you need to configure the distribution and remote routers to use EIGRP and to configure only the device as a stub. Only specified routes are propagated from the device. The device responds to all queries for summaries, connected routes, and routing updates.

Any neighbor that receives a packet informing it of the stub status does not query the stub router for any routes, and a router that has a stub peer does not query that peer. The stub router depends on the distribution router to send the proper updates to all peers.

In the figure given below, device B is configured as an EIGRP stub router. Devicees A and C are connected to the rest of the WAN. Device B advertises connected, static, redistribution, and summary routes to Device A and C. Device B does not advertise any routes learned from Device A (and the reverse).

Figure 5: EIGRP Stub Router Configuration



How to Configure EIGRP

To create an EIGRP routing process, you must enable EIGRP and associate networks. EIGRP sends updates to the interfaces in the specified networks. If you do not specify an interface network, it is not advertised in any EIGRP update.

Note If you have routers on your network that are configured for IGRP, and you want to change to EIGRP, you must designate transition routers that have both IGRP and EIGRP configured. In these cases, perform Steps 1 through 3 in the next section and also see the "Configuring Split Horizon" section. You must use the same AS number for routes to be automatically redistributed.

Default EIGRP Configuration

Table 16: Default EIGRP Configuration

Feature	Default Setting
Auto summary	Disabled.
Default-information	Exterior routes are accepted and default information is passed betwee processes when doing redistribution.

Feature	Default Setting
Default metric	Only connected routes and interface static routes can be redistribu default metric. The metric includes: • Bandwidth: 0 or greater kb/s. • Delay (tens of microseconds): 0 or any positive number that is 39.1 nanoseconds. • Reliability: any number between 0 and 255 (255 means 100 p reliability). • Loading: effective bandwidth as a number between 0 and 255 percent loading). • MTU: maximum transmission unit size of the route in bytes. 0 c integer.
Distance	Internal distance: 90. External distance: 170.
EIGRP log-neighbor changes	Disabled. No adjacency changes logged.
IP authentication key-chain	No authentication provided.
IP authentication mode	No authentication provided.
IP bandwidth-percent	50 percent.
IP hello interval	For low-speed nonbroadcast multiaccess (NBMA) networks: 60 sec networks: 5 seconds.
IP hold-time	For low-speed NBMA networks: 180 seconds; all other networks:
IP split-horizon	Enabled.
IP summary address	No summary aggregate addresses are predefined.
Metric weights	tos: 0; k1 and k3: 1; k2, k4, and k5: 0
Network	None specified.
Nonstop Forwarding (NSF) Awareness	Enabled for IPv4 on switches running the Allows Layer 3 switche forwarding packets from a neighboring NSF-capable router during software changes.
NSF capability	Disabled. **Note** The Device supports EIGRP NSF-capable routing for
Offset-list	Disabled.
Router EIGRP	Disabled.
Set metric	No metric set in the route map.

Feature	Default Setting
Traffic-share	Distributed proportionately to the ratios of the metrics.
Variance	1 (equal-cost load-balancing).

Configuring Basic EIGRP Parameters

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router eigrp** autonomous-system **Example:** `Device(config)# router eigrp 10`	Enables an EIGRP routing process, and enter router configuration mode. The AS number identifies the routes to other EIGRP routers and is used to tag routing information.
Step 3	**nsf** **Example:** `Device(config-router)# nsf`	(Optional) Enables EIGRP NSF. Enter this command on the active switch and on all of its peers.
Step 4	**network** network-number **Example:** `Device(config-router)# network 192.168.0.0`	Associate networks with an EIGRP routing process. EIGRP sends updates to the interfaces in the specified networks.
Step 5	**eigrp log-neighbor-changes** **Example:** `Device(config-router)# eigrp log-neighbor-changes`	(Optional) Enables logging of EIGRP neighbor changes to monitor routing system stability.
Step 6	**metric weights** *tos k1 k2 k3 k4 k5* **Example:** `Device(config-router)# metric weights 0 2 0 2 0 0`	(Optional) Adjust the EIGRP metric. Although the defaults have been carefully set to provide excellent operation in most networks, you can adjust them. **Caution** Setting metrics is complex and is not recommended without guidance from an experienced network designer.
Step 7	**offset-list** [*access-list number* | *name*] {**in** | **out**} *offset* [*type number*]	(Optional) Applies an offset list to routing metrics to increase incoming and outgoing metrics to routes learned

	Command or Action	Purpose
	Example: `Device(config-router)# offset-list 21 out 10`	through EIGRP. You can limit the offset list with an access list or an interface.
Step 8	**auto-summary** **Example:** `Device(config-router)# auto-summary`	(Optional) Enables automatic summarization of subnet routes into network-level routes.
Step 9	**interface** *interface-id* **Example:** `Device(config-router)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 10	**ip summary-address eigrp** *autonomous-system-number address mask* **Example:** `Device(config-if)# ip summary-address eigrp 1 192.168.0.0 255.255.0.0`	(Optional) Configures a summary aggregate.
Step 11	**end** **Example:** `Device(config-if)#end`	Returns to privileged EXEC mode.
Step 12	**show ip protocols** **Example:** `Device# show ip protocols`	Verifies your entries. For NSF awareness, the output shows: *** IP Routing is NSF aware *** EIGRP NSF enabled
Step 13	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring EIGRP Interfaces

Other optional EIGRP parameters can be configured on an interface basis.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 3	**ip bandwidth-percent eigrp** *percent* **Example:** `Device(config-if)# ip bandwidth-percent eigrp 60`	(Optional) Configures the percentage of bandwidth that can be used by EIGRP on an interface. The default is 50 percent.
Step 4	**ip summary-address eigrp** *autonomous-system-number address mask* **Example:** `Device(config-if)# ip summary-address eigrp 109 192.161.0.0 255.255.0.0`	(Optional) Configures a summary aggregate address for a specified interface (not usually necessary if auto-summary is enabled).
Step 5	**ip hello-interval eigrp** *autonomous-system-number seconds* **Example:** `Device(config-if)# ip hello-interval eigrp 109 10`	(Optional) Change the hello time interval for an EIGRP routing process. The range is 1 to 65535 seconds. The default is 60 seconds for low-speed NBMA networks and 5 seconds for all other networks.
Step 6	**ip hold-time eigrp** *autonomous-system-number seconds* **Example:** `Device(config-if)# ip hold-time eigrp 109 40`	(Optional) Change the hold time interval for an EIGRP routing process. The range is 1 to 65535 seconds. The default is 180 seconds for low-speed NBMA networks and 15 seconds for all other networks. **Caution** Do not adjust the hold time without consulting Cisco technical support.
Step 7	**no ip split-horizon eigrp** *autonomous-system-number* **Example:** `Device(config-if)# no ip split-horizon eigrp 109`	(Optional) Disables split horizon to allow route information to be advertised by a router out any interface from which that information originated.
Step 8	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)# end`	
Step 9	**show ip eigrp interface** **Example:** `Device# show ip eigrp interface`	Displays which interfaces EIGRP is active on and information about EIGRP relating to those interfaces.
Step 10	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring EIGRP Route Authentication

EIGRP route authentication provides MD5 authentication of routing updates from the EIGRP routing protocol to prevent the introduction of unauthorized or false routing messages from unapproved sources.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**interface** interface-id **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 3	**ip authentication mode eigrp** *autonomous-system* **md5** **Example:** `Device(config-if)# ip authentication mode eigrp 104 md5`	Enables MD5 authentication in IP EIGRP packets.
Step 4	**ip authentication key-chain eigrp** *autonomous-system key-chain* **Example:** `Device(config-if)# ip authentication key-chain eigrp 105 chain1`	Enables authentication of IP EIGRP packets.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 6	**key chain** *name-of-chain* **Example:** `Device(config)# key chain chain1`	Identify a key chain and enter key-chain configuration mode. Match the name configured in Step 4.
Step 7	**key** *number* **Example:** `Device(config-keychain)# key 1`	In key-chain configuration mode, identify the key number.
Step 8	**key-string** *text* **Example:** `Device(config-keychain-key)# key-string key1`	In key-chain key configuration mode, identify the key string.
Step 9	**accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain-key)# accept-lifetime 13:30:00 Jan 25 2011 duration 7200`	(Optional) Specifies the time period during which the key can be received. The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 10	**send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain-key)# send-lifetime 14:00:00 Jan 25 2011 duration 3600`	(Optional) Specifies the time period during which the key can be sent. The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 11	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 12	**show key chain** **Example:** `Device# show key chain`	Displays authentication key information.

	Command or Action	Purpose
Step 13	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring and Maintaining EIGRP

You can delete neighbors from the neighbor table. You can also display various EIGRP routing statistics. The table given below lists the privileged EXEC commands for deleting neighbors and displaying statistics.

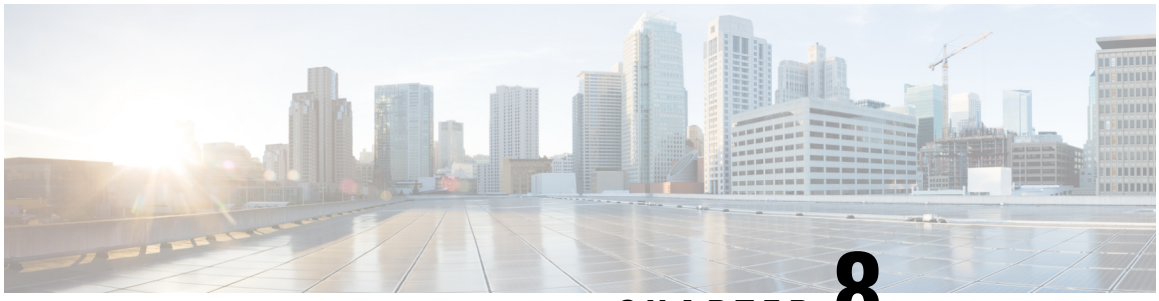
Table 17: IP EIGRP Clear and Show Commands

clear ip eigrp neighbors [*if-address* | *interface*]	Deletes neighbors from the neigh
show ip eigrp interface [*interface*] [*as number*]	Displays information about interf
show ip eigrp neighbors [*type-number*]	Displays EIGRP discovered neig
show ip eigrp topology [*autonomous-system-number*] | [[*ip-address*] *mask*]]	Displays the EIGRP topology tab
show ip eigrp traffic [*autonomous-system-number*]	Displays the number of packets s process.

Feature Information for EIGRP

Table 18: Feature Information for EIGRP

Feature Name	Release	Feature Information
EIGRP (Enhanced IGRP)	Cisco IOS XE Everest 16.6.1	This feature was introduced.



CHAPTER 8

BFD - EIGRP Support

The BFD-EIGRP Support feature configures the Enhanced Interior Gateway Routing Protocol (EIGRP) with Bidirectional Forwarding Detection (BFD) so that EIGRP registers with BFD and receives all forwarding path detection failure messages from BFD.

Prerequisites for BFD-EIGRP Support

- Enhanced Interior Gateway Routing Protocol (EIGRP) must be running on all participating routers.
- The baseline parameters for Bidirectional Forwarding Detection (BFD) sessions on the interfaces over which you want to run BFD sessions to BFD neighbors must be configured using the **bfd** command.

Information About BFD - EIGRP Support

Overview of BFD-EIGRP Support

The BFD-EIGRP Support feature configures Bidirectional Forwarding Detection (BFD) feature for Enhanced Interior Gateway Routing Protocol (EIGRP) so that EIGRP registers with the BFD sessions on the routing interfaces, and receives forwarding path detection failure messages from BFD.

Use **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* command to enable BFD on any interface. Use the **bfd all-interfaces** command in router configuration mode to enable BFD for all of the interfaces where EIGRP routing is enabled. Use the **bfd interface** *type number* command in router configuration mode to enable BFD for a subset of the interfaces where EIGRP routing is enabled.

How to Configure BFD - EIGRP Support

Configuring BFD - EIGRP Support

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router eigrp** *as-number*
4. Do one of the following:
   - **bfd all-interfaces**
   - **bfd interface** *type number*
5. **end**
6. **show bfd neighbors** [**details**]
7. **show ip eigrp interfaces** [*type number*] [*as-number*] [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router eigrp** *as-number* **Example:** `Device(config)# router eigrp 123`	Configures the EIGRP routing process and enters router configuration mode.

Step 4	Do one of the following:
	• **bfd all-interfaces**
	• **bfd interface** *type number*
	Example:
	`Device(config-router)# bfd all-interfaces`
	Example:
	`Device(config-router)# bfd interface FastEthernet 6/0`
		Enables BFD globally on all interfaces associated with the EIGRP routing process.
		or
		Enables BFD on a per-interface basis for one or more interfaces associated with the EIGRP routing process.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns the router to privileged EXEC mode.
Step 6	**show bfd neighbors** [**details**] **Example:** `Device# show bfd neighbors details`	(Optional) Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered.
Step 7	**show ip eigrp interfaces** [*type number*] [*as-number*] [**detail**] **Example:** `Device# show ip eigrp interfaces detail`	(Optional) Displays the interfaces for which BFD support for EIGRP has been enabled.

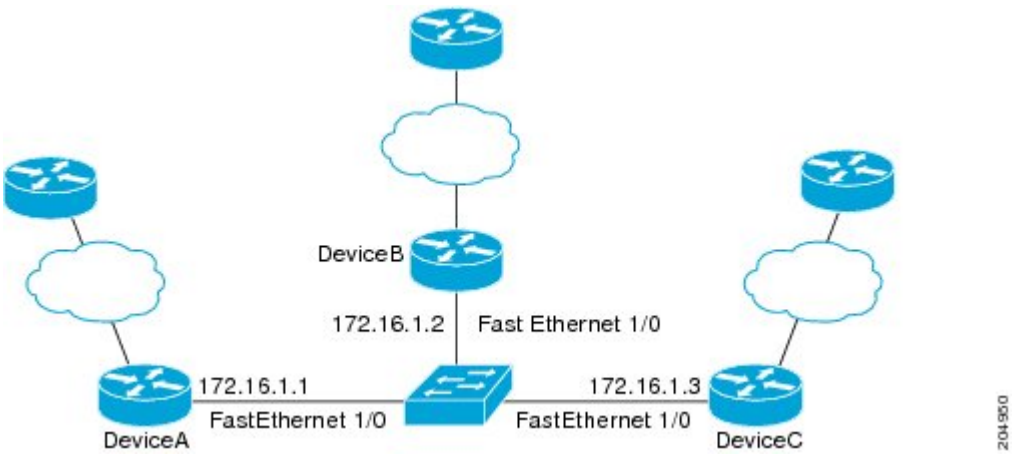
Configuration Examples for BFD - EIGRP Support

Example: Configuring BFD in an EIGRP Network with Echo Mode Enabled by Default

In the following example, the EIGRP network contains DeviceA, DeviceB, and DeviceC. Fast Ethernet interface 1/0 on DeviceA is connected to the same network as Fast Ethernet interface 1/0 on Device B. Fast Ethernet interface 1/0 on DeviceB is connected to the same network as Fast Ethernet interface 1/0 on DeviceC.

DeviceA and DeviceB are running BFD Version 1, which supports echo mode, and DeviceC is running BFD Version 0, which does not support echo mode. The BFD sessions between DeviceC and its BFD neighbors are said to be running echo mode with asymmetry because echo mode will run on the forwarding path for DeviceA and DeviceB, and their echo packets will return along the same path for BFD sessions and failure detections, while their BFD neighbor DeviceC runs BFD Version 0 and uses BFD controls packets for BFD sessions and failure detections.

The figure below shows a large EIGRP network with several devices, three of which are BFD neighbors that are running EIGRP as their routing protocol.



The example, starting in global configuration mode, shows the configuration of BFD.

Configuration for DeviceA

```
interface Fast Ethernet0/0
no shutdown
ip address 10.4.9.14 255.255.255.0
duplex auto
speed auto
!
interface Fast Ethernet1/0
ip address 172.16.1.1 255.255.255.0
bfd interval 50 min_rx 50 multiplier 3
no shutdown
duplex auto
speed auto
!
router eigrp 11
network 172.16.0.0
bfd all-interfaces
auto-summary
!
ip default-gateway 10.4.9.1
ip default-network 0.0.0.0
ip route 0.0.0.0 0.0.0.0 10.4.9.1
ip route 172.16.1.129 255.255.255.255 10.4.9.1
!
no ip http server
!
logging alarm informational
!
control-plane
!
line con 0
exec-timeout 30 0
stopbits 1
line aux 0
stopbits 1
line vty 0 4
login
!
!
end
```

Configuration for DeviceB

```
!
interface Fast Ethernet0/0
no shutdown
ip address 10.4.9.34 255.255.255.0
duplex auto
speed auto
!
interface Fast Ethernet1/0
ip address 172.16.1.2 255.255.255.0
bfd interval 50 min_rx 50 multiplier 3
no shtdown
duplex auto
speed auto
!
router eigrp 11
network 172.16.0.0
bfd all-interfaces
auto-summary
!
ip default-gateway 10.4.9.1
ip default-network 0.0.0.0
ip route 0.0.0.0 0.0.0.0 10.4.9.1
ip route 172.16.1.129 255.255.255.255 10.4.9.1
!
no ip http server
!
logging alarm informational
!
control-plane
!
line con 0
exec-timeout 30 0
stopbits 1
line aux 0
stopbits 1
line vty 0 4
login
!
!
end
```

Configuration for DeviceC

```
!
!
interface Fast Ethernet0/0
no shutdown
ip address 10.4.9.34 255.255.255.0
duplex auto
speed auto
!
interface Fast Ethernet1/0
ip address 172.16.1.2 255.255.255.0
bfd interval 50 min_rx 50 multiplier 3
no shutdown
duplex auto
speed auto
!
router eigrp 11
network 172.16.0.0
```

```
bfd all-interfaces
auto-summary
!
ip default-gateway 10.4.9.1
ip default-network 0.0.0.0
ip route 0.0.0.0 0.0.0.0 10.4.9.1
ip route 172.16.1.129 255.255.255.255 10.4.9.1
!
no ip http server
!
logging alarm informational
!
control-plane
!
line con 0
exec-timeout 30 0
stopbits 1
line aux 0
stopbits 1
line vty 0 4
login
!
!
end
```

The output from the **show bfd neighbors details** command from DeviceA verifies that BFD sessions are created among all three devices and that EIGRP is registered for BFD support. The first group of output shows that DeviceC with the IP address 172.16.1.3 runs BFD Version 0 and therefore does not use the echo mode. The second group of output shows that DeviceB with the IP address 172.16.1.2 runs BFD Version 1, and the 50 millisecond BFD interval parameter had been adopted. The relevant command output is shown in bold in the output.

```
DeviceA# show bfd neighbors details

OurAddr
       NeighAddr
      LD/RD  RH/RS    Holdown(mult)  State     Int
172.16.1.1    172.16.1.3
     5/3    1(RH)    150 (3 )       Up   Fa1/0
Session state is UP and not using echo function.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 50000, MinRxInt: 50000, Multiplier: 3
Received MinRxInt: 50000, Received Multiplier: 3
Holdown (hits): 150(0), Hello (hits): 50(1364284)
Rx Count: 1351813, Rx Interval (ms) min/max/avg: 28/64/49 last: 4 ms ago
Tx Count: 1364289, Tx Interval (ms) min/max/avg: 40/68/49 last: 32 ms ago
Registered protocols: EIGRP
Uptime: 18:42:45
Last packet: Version: 0
            - Diagnostic: 0
             I Hear You bit: 1     - Demand bit: 0
             Poll bit: 0           - Final bit: 0
             Multiplier: 3         - Length: 24
             My Discr.: 3          - Your Discr.: 5
             Min tx interval: 50000    - Min rx interval: 50000
             Min Echo interval: 0
OurAddr       NeighAddr
     LD/RD  RH/RS   Holdown(mult)  State     Int
172.16.1.1    172.16.1.2

    6/1    Up        0    (3 )   Up        Fa1/0
```

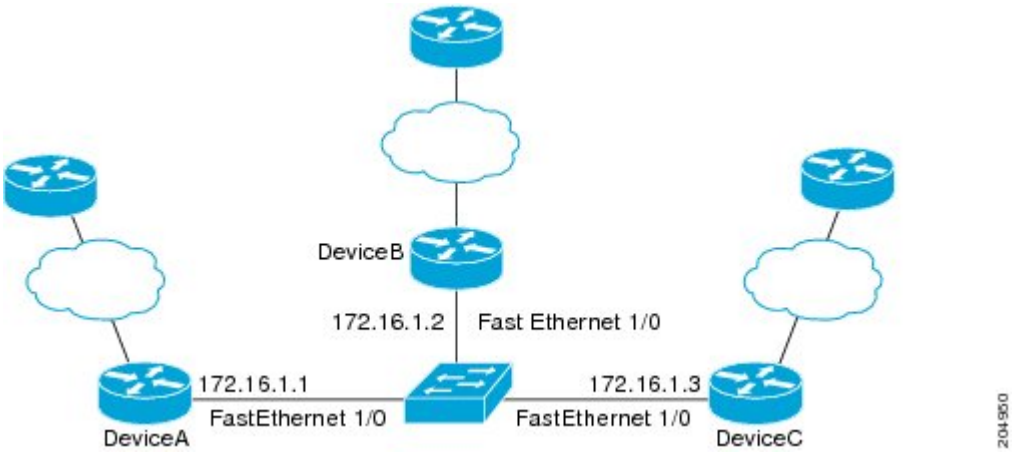
```
Session state is UP and using echo function with 50 ms interval.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 1000000, MinRxInt: 1000000, Multiplier: 3
Received MinRxInt: 1000000, Received Multiplier: 3
Holdown (hits): 3000(0), Hello (hits): 1000(317)
Rx Count: 305, Rx Interval (ms) min/max/avg: 1/1016/887 last: 448 ms ago
Tx Count: 319, Tx Interval (ms) min/max/avg: 1/1008/880 last: 532 ms ago
Registered protocols: EIGRP
Uptime: 00:04:30
Last packet: Version: 1

         - Diagnostic: 0
             State bit: Up         - Demand bit: 0
             Poll bit: 0           - Final bit: 0
             Multiplier: 3         - Length: 24
             My Discr.: 1          - Your Discr.: 6
             Min tx interval: 1000000    - Min rx interval: 1000000
             Min Echo interval: 50000
```

The output from the **show bfd neighbors details** command on Device B verifies that BFD sessions have been created and that EIGRP is registered for BFD support. As previously noted, DeviceA runs BFD Version 1, therefore echo mode is running, and DeviceC runs BFD Version 0, so echo mode does not run. The relevant command output is shown in bold in the output.

```
DeviceB# show bfd neighbors details

OurAddr       NeighAddr
     LD/RD  RH/RS   Holdown(mult)  State     Int
172.16.1.2    172.16.1.1
    1/6    Up        0    (3 )   Up        Fa1/0
Session state is UP and using echo function with 50 ms interval.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 1000000, MinRxInt: 1000000, Multiplier: 3
Received MinRxInt: 1000000, Received Multiplier: 3
Holdown (hits): 3000(0), Hello (hits): 1000(337)
Rx Count: 341, Rx Interval (ms) min/max/avg: 1/1008/882 last: 364 ms ago
Tx Count: 339, Tx Interval (ms) min/max/avg: 1/1016/886 last: 632 ms ago
Registered protocols: EIGRP
Uptime: 00:05:00
Last packet: Version: 1
           - Diagnostic: 0
             State bit: Up         - Demand bit: 0
             Poll bit: 0           - Final bit: 0
             Multiplier: 3         - Length: 24
             My Discr.: 6          - Your Discr.: 1
             Min tx interval: 1000000    - Min rx interval: 1000000
             Min Echo interval: 50000
OurAddr       NeighAddr

 LD/RD  RH/RS   Holdown(mult)  State     Int
172.16.1.2    172.16.1.3
     3/6    1(RH)     118  (3 )   Up        Fa1/0
Session state is UP and not using echo function.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 50000, MinRxInt: 50000, Multiplier: 3
Received MinRxInt: 50000, Received Multiplier: 3
Holdown (hits): 150(0), Hello (hits): 50(5735)
Rx Count: 5731, Rx Interval (ms) min/max/avg: 32/72/49 last: 32 ms ago
Tx Count: 5740, Tx Interval (ms) min/max/avg: 40/64/50 last: 44 ms ago
Registered protocols: EIGRP
Uptime: 00:04:45
Last packet: Version: 0
```

```
- Diagnostic: 0
 I Hear You bit: 1      - Demand bit: 0
 Poll bit: 0            - Final bit: 0
 Multiplier: 3          - Length: 24
 My Discr.: 6           - Your Discr.: 3
 Min tx interval: 50000    - Min rx interval: 50000
 Min Echo interval: 0
```

The figure below shows that Fast Ethernet interface 1/0 on DeviceB has failed. When Fast Ethernet interface 1/0 on DeviceB is shut down, the BFD statistics of the corresponding BFD sessions on DeviceA and DeviceB are reduced.



When Fast Ethernet interface 1/0 on DeviceB fails, BFD will no longer detect Device B as a BFD neighbor for DeviceA or for DeviceC. In this example, Fast Ethernet interface 1/0 has been administratively shut down on DeviceB.

The following output from the **show bfd neighbors** command on DeviceA now shows only one BFD neighbor for DeviceA in the EIGRP network. The relevant command output is shown in bold in the output.

```
DeviceA# show bfd neighbors
OurAddr       NeighAddr

    LD/RD  RH/RS   Holdown(mult)  State     Int
172.16.1.1    172.16.1.3

    5/3    1(RH)     134  (3 )   Up        Fa1/0
```

The following output from the **show bfd neighbors** command on DeviceC also now shows only one BFD neighbor for DeviceC in the EIGRP network. The relevant command output is shown in bold in the output.

```
DeviceC# show bfd neighbors

OurAddr       NeighAddr

   LD/RD RH  Holdown(mult)  State     Int
172.16.1.3    172.16.1.1

    3/5  1   114  (3 )      Up        Fa1/0
```

Additional References for BFD-EIGRP Support

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	See the *Command Reference (Catalyst 9400 Series Switches)*

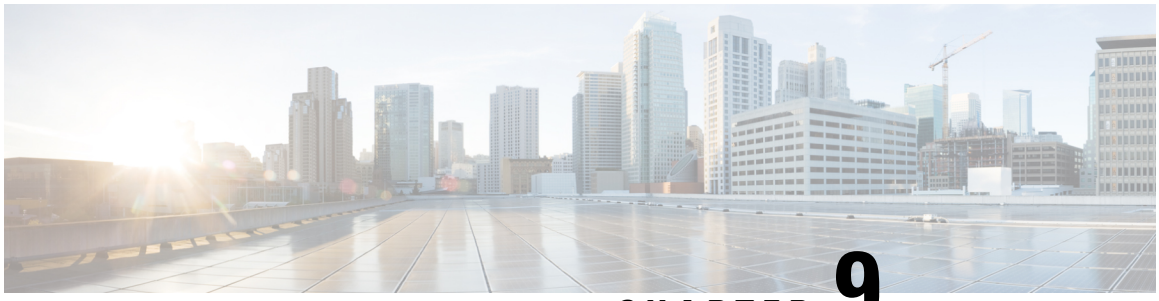
Feature Information for BFD-EIGRP Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Table 19: Feature Information for BFD-EIGRP Support

Feature Name	Releases	Feature Information
BFD-EIGRP Support	Cisco IOS XE Everest 16.6.2	The BFD-EIGRP Support feature configures the Enhanced Interior Gateway Routing Protocol (EIGRP) with Bidirectional Forwarding Detection (BFD) so that EIGRP registers with BFD and receives all forwarding path detection failure messages from BFD. In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.



CHAPTER 9

BFD - Static Route Support

The BFD - Static Route Support feature enables association of static routes with a static Bidirectional Forwarding Detection (BFD) configuration in order to monitor static route reachability using the configured BFD session. Depending on status of the BFD session, static routes are added to or removed from the Routing Information Base (RIB).

Prerequisites for BFD - Static Route Support

- Cisco Express Forwarding and IP routing must be enabled on all participating devices.
- The baseline parameters for Bidirectional Forwarding Detection (BFD) sessions on the interfaces over which you want to run BFD sessions to BFD neighbors must be configured.

Restrictions for BFD - Static Route Support

- The configuration of BFD on virtual-template and dialer interfaces is incorrectly allowed by the software; however, BFD functionality on virtual-template and dialer interfaces is not supported. Avoid configuring BFD on virtual-template and dialer interfaces.
- BFD works only for directly connected neighbors. BFD neighbors must be no more than one IP hop away. Multihop configurations are not supported.

Information About BFD - Static Route Support

Overview of BFD - Static Route Support

The BFD - Static Route Support feature enables association of static routes with a static Bidirectional Forwarding Detection (BFD) configuration in order to monitor static route reachability using the configured BFD session. Depending on status of the BFD session, static routes are added to or removed from the Routing Information Base (RIB).

Unlike dynamic routing protocols, such as OSPF and BGP, static routing has no method of peer discovery. Therefore, when BFD is configured, the reachability of the gateway is completely dependent on the state of the BFD session to the specified neighbor. Unless the BFD session is up, the gateway for the static route is considered unreachable, and therefore the affected routes will not be installed in the appropriate RIB.

For a BFD session to be successfully established, BFD must be configured on the interface on the peer and there must be a BFD client registered on the peer for the address of the BFD neighbor. When an interface is used by dynamic routing protocols, the latter requirement is usually met by configuring the routing protocol instances on each neighbor for BFD. When an interface is used exclusively for static routing, this requirement must be met by configuring static routes on the peers.

If a BFD configuration is removed from the remote peer while the BFD session is in the up state, the updated state of the BFD session is not signaled to IPv4 static. This will cause the static route to remain in the RIB. The only workaround is to remove the IPv4 static BFD neighbor configuration so that the static route no longer tracks BFD session state. Also, if you change the encapsulation type on a serial interface to one that is unsupported by BFD, BFD will be in a down state on that interface. The workaround is to shut down the interface, change to a supported encapsulation type, and then reconfigure BFD.

A single BFD session can be used by an IPv4 static client to track the reachability of next hops through a specific interface. You can assign a BFD group for a set of BFD-tracked static routes. Each group must have one active static BFD configuration, one or more passive BFD configurations, and the corresponding static routes to be BFD-tracked. Nongroup entries are BFD-tracked static routes for which a BFD group is not assigned. A BFD group must accommodate static BFD configurations that can be part of different VRFs. Effectively, the passive static BFD configurations need not be in the same VRF as that of the active configuration.

For each BFD group, there can be only one active static BFD session. You can configure the active BFD session by adding a static BFD configuration and a corresponding static route that uses the BFD configuration. The BFD session in a group is created only when there is an active static BFD configuration and the static route that uses the static BFD configuration. When the active static BFD configuration or the active static route is removed from a BFD group, all the passive static routes are withdrawn from the RIB. Effectively, all the passive static routes are inactive until an active static BFD configuration and a static route to be tracked by the active BFD session are configured in the group.

Similarly, for each BFD group, there can be one or more passive static BFD configurations and their corresponding static routes to be BFD-tracked. Passive static session routes take effect only when the active BFD session state is reachable. Though the active BFD session state of the group is reachable, the passive static route is added to the RIB only if the corresponding interface state is up. When a passive BFD session is removed from a group, it will not affect the active BFD session if one existed, or the BFD group reachability status.

How to Configure BFD - Static Route Support

Configuring BFD - EIGRP Support

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router eigrp** *as-number*
4. Do one of the following:
   - **bfd all-interfaces**
   - **bfd interface** *type number*
5. **end**
6. **show bfd neighbors** [**details**]
7. **show ip eigrp interfaces** [*type number*] [*as-number*] [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router eigrp** *as-number* **Example:** `Device(config)# router eigrp 123`	Configures the EIGRP routing process and enters router configuration mode.

Step 4	Do one of the following:
	• **bfd all-interfaces**
	• **bfd interface** *type number*
	Example:
	`Device(config-router)# bfd all-interfaces`
	Example:
	`Device(config-router)# bfd interface FastEthernet 6/0`
		Enables BFD globally on all interfaces associated with the EIGRP routing process.
		or
		Enables BFD on a per-interface basis for one or more interfaces associated with the EIGRP routing process.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns the router to privileged EXEC mode.
Step 6	**show bfd neighbors** [**details**] **Example:** `Device# show bfd neighbors details`	(Optional) Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered.
Step 7	**show ip eigrp interfaces** [*type number*] [*as-number*] [**detail**] **Example:** `Device# show ip eigrp interfaces detail`	(Optional) Displays the interfaces for which BFD support for EIGRP has been enabled.

Configuration Examples for BFD - Static Route Support

Example: Configuring BFD - Static Route Support

In the following example, the network consists of Device A and Device B. Serial interface 2/0 on Device A is connected to the same network as serial interface 2/0 on Device B. In order for the BFD session to come up, Device B must be configured.

Device A

```
configure terminal
 interface Serial 2/0
  ip address 10.201.201.1 255.255.255.0
  bfd interval 500 min_rx 500 multiplier 5
  ip route static bfd Serial 2/0 10.201.201.2
  ip route 10.0.0.0 255.0.0.0 Serial 2/0 10.201.201.2
```

Device B

```
configure terminal
 interface Serial 2/0
  ip address 10.201.201.2 255.255.255.0
  bfd interval 500 min_rx 500 multiplier 5
  ip route static bfd Serial 2/0 10.201.201.1
  ip route 10.1.1.1 255.255.255.255 Serial 2/0 10.201.201.1
```

Note that the static route on Device B exists solely to enable the BFD session between 10.201.201.1 and 10.201.201.2. If there is no useful static route that needs to be configured, select a prefix that will not affect packet forwarding, for example, the address of a locally configured loopback interface.

In the following example, there is an active static BFD configuration to reach 209.165.200.225 through Ethernet interface 0/0 in the BFD group testgroup. As soon as the static route is configured that is tracked by

the configured static BFD, a single hop BFD session is initiated to 209.165.200.225 through Ethernet interface 0/0. The prefix 10.0.0.0/8 is added to the RIB if a BFD session is successfully established.

```
configure terminal
 ip route static bfd Ethernet 0/0 209.165.200.225 group testgroup
 ip route 10.0.0.0 255.255.255.224 Ethernet 0/0 209.165.200.225
```

In the following example, a BFD session to 209.165.200.226 through Ethernet interface 0/0.1001 is marked to use the group testgroup. That is, this configuration is a passive static BFD. Though there are static routes to be tracked by the second static BFD configuration, a BFD session is not triggered for 209.165.200.226 through Ethernet interface 0/0.1001. The existence of the prefixes 10.1.1.1/8 and 10.2.2.2/8 is controlled by the active static BFD session (Ethernet interface 0/0 209.165.200.225).

```
configure terminal
 ip route static bfd Ethernet 0/0 209.165.200.225 group testgroup
 ip route 10.0.0.0 255.255.255.224 Ethernet 0/0 209.165.200.225
 ip route static bfd Ethernet 0/0.1001 209.165.200.226 group testgroup passive
 ip route 10.1.1.1 255.255.255.224 Ethernet 0/0.1001 209.165.200.226
 ip route 10.2.2.2 255.255.255.224 Ethernet 0/0.1001 209.165.200.226
```

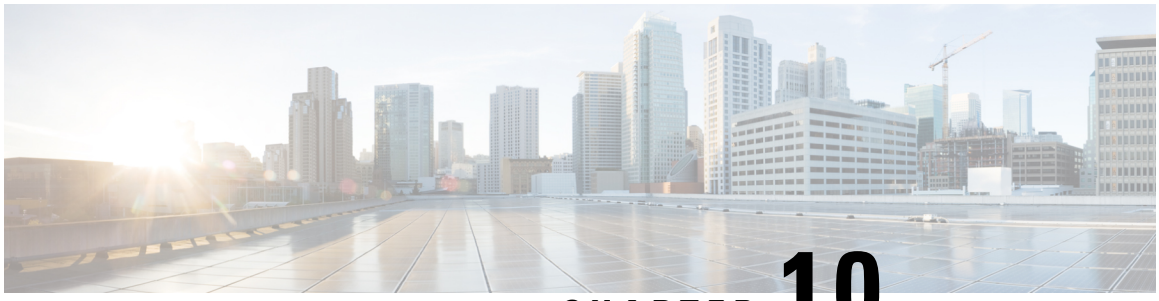
Feature Information for BFD - Static Route Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Table 20: Feature Information for BFD - Static Route Support

Feature Name	Releases	Feature Information
BFD - Static Route Support	Cisco IOS XE Everest 16.6.2	The BFD - Static Route Support feature enables association of static routes with a static Bidirectional Forwarding Detection (BFD) configuration in order to monitor static route reachability using the configured BFD session. Depending on status of the BFD session, static routes are added to or removed from the Routing Information Base (RIB). In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.



CHAPTER 10

BFD - VRF Support

The BFD - VRF Support feature enables Bidirectional Forwarding Detection (BFD) support for Virtual Routing and Forwarding (VRF) on Provider Edge (PE) and Customer Edge (CE) devices to provide fast detection of routing protocol failures between the devices.

Prerequisites for BFD - VRF Support

All Bidirectional Forwarding Detection (BFD) clients must be Virtual Routing and Forwarding (VRF)-aware.

Information About BFD - VRF Support

Overview of BFD - VRF Support

The BFD - VRF Support feature enables Bidirectional Forwarding Detection (BFD) support for Virtual Routing and Forwarding (VRF) on Provider Edge (PE) and Customer Edge (CE) devices to provide fast detection of routing protocol failures between the devices.

A BFD client establishes a Virtual Private Networking (VPN) session with devices that have BFD configured on them before requesting for session monitoring. However, there are no route lookups to determine whether a BFD neighbor is connected to the same VPN session or a different one. BFD relies on its client to get information about the VPN session to monitor the associated neighbor device. All information about VPN sessions is used to forward BFD control packets to the appropriate VPN through Cisco Express Forwarding (CEF).

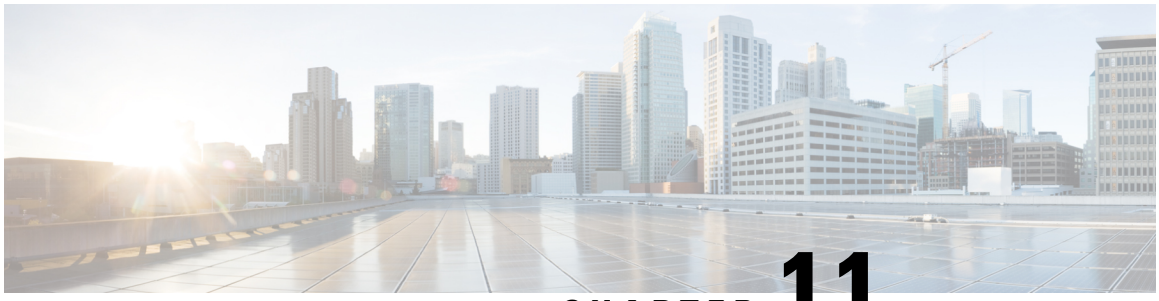
Feature Information for BFD - VRF Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Table 21: Feature Information for BFD - VRF Support

Feature Name	Releases	Feature Information
BFD - VRF Support	Cisco IOS XE Everest 16.6.2	The BFD - VRF Support feature enables BFD support for VRFs on PE and CE devices to provide fast detection of routing protocol failures between the devices. In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.



CHAPTER 11

BFD IPv6 Encapsulation Support

Bidirectional Forwarding Detection for IPv6 encapsulations are described within a session information structure. These session information structures are defined by BFDv6 for the protocols supported. BFDv6 uses information from the session information structures to determine the correct encapsulation for BFDv6 packets on that session.

Prerequisites for BFD IPv6 Encapsulation Support

- When using Bidirectional Forwarding Detection over IPv6 (BFDv6), IPv6 Cisco Express Forwarding and IPv6 unicast routing must be enabled on all participating routers.
- When you configure BFD IPv6 software sessions, you should configure the following CLI command:

  no ipv6 nd nud igp

Restrictions for BFD IPv6 Encapsulation Support

- BFDv6 supports only global IPv6 neighbor addresses if a global IPv6 address is configured on the interface.
- Only asynchronous mode is supported. In asynchronous mode, either BFDv6 peer can initiate a BFDv6 session.

Information About BFD IPv6 Encapsulation Support

Overview of the BFDv6 Protocol

This section describes the BFDv6 protocol, how it is different from BFD for IPv4, and how it works with BFD for IPv4. BFD is a detection protocol designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols. In addition to fast forwarding path failure detection, BFD provides a consistent failure detection method for network administrators. BFDv6 provides IPv6 support by accommodating IPv6 addresses and provides the ability to create BFDv6 sessions.

BFDv6 Registration

BFD clients register with BFD using a registry application program interface (API). The registry arguments include protocol type and the address and interface description block (IDB) of the route to be monitored. These APIs and arguments are all assumed by BFD to be IPv4.

BFDv6 has registries from which these arguments have been removed, and the protocol and encapsulation are described within a session information structure. These session information structures are defined by BFDv6 for the protocols supported. BFDv6 uses information from the session information structures to determine the correct encapsulation for BFDv6 packets on that session.

BFDv6 Global and Link-Local Addresses

BFDv6 supports both global and link-local IPv6 addresses for neighbor creation. BFDv6 sessions select source addresses to match the neighbor address types (for example, global IPv6 address neighbors must be paired with global IPv6 source addresses and link-local IPv6 address neighbors must be paired with link-local IPv6 source addresses). The table below shows the address pairings that BFDv6 supports.

Table 22: BFDv6 Address Pairings for Neighbor Creation

Source Address	Destination Address	Status
Global	Global	Supported
Global	Link local	Not supported
Link local	Global	Not supported
Link local	Link local	Supported

Because all IPv6-enabled interfaces have a link-local address and BFDv6 selects the source address, link-local address neighbors are always paired with a link-local interface address. The link-local source address with global destination address is not supported by Cisco Express Forwarding. Therefore, a global IPv6 address must be configured on an interface before a session with a global address neighbor may be established in BFDv6. BFDv6 rejects any sessions in which the neighbor address is global and no global address is configured on the interface.

Note The behavior of a unique local address (ULA) in BFDv6 is the same as a global address.

BFD for IPv4 and IPv6 on the Same Interface

BFD supports multiple IPv4 and IPv6 sessions per interface, with no restriction on the protocol of those sessions.

How to Configure BFD IPv6 Encapsulation Support

Configuring Baseline BFD Session Parameters on the Interface

Repeat this task for each interface over which you want to run BFD sessions to BFD neighbors.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/0`	Specifies an interface type and number, and places the device in interface configuration mode.
Step 4	**bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* **Example:**	Enables BFD on the interface.

	Command or Action	Purpose
	`Device(config-if)# bfd interval 50 min_rx 50 multiplier 5`	

Configuration Examples for BFD IPv6 Encapsulation Support

Example: Configuring BFD Session Parameters on the Interface

```
Device# show ipv6 ospf neighbor detail

 Neighbor 172.16.4.4
    In the area 0 via interface POS4/0
    Neighbor: interface-id 14, link-local address FE80::205:5FFF:FED3:5406
    Neighbor priority is 1, State is FULL, 6 state changes
    Options is 0x63AD1B0D
    Dead timer due in 00:00:33
    Neighbor is up for 00:48:56
    Index 1/1/1, retransmission queue length 0, number of retransmission 1
    First 0x0(0)/0x0(0)/0x0(0) Next 0x0(0)/0x0(0)/0x0(0)
    Last retransmission scan length is 1, maximum is 1
    Last retransmission scan time is 0 msec, maximum is 0 msec
 Neighbor 172.16.3.3
    In the area 1 via interface FastEthernet0/0
    Neighbor: interface-id 3, link-local address FE80::205:5FFF:FED3:5808
    Neighbor priority is 1, State is FULL, 6 state changes
    DR is 172.16.6.6 BDR is 172.16.3.3
    Options is 0x63F813E9
    Dead timer due in 00:00:33
    Neighbor is up for 00:09:00
    Index 1/1/2, retransmission queue length 0, number of retransmission 2
    First 0x0(0)/0x0(0)/0x0(0) Next 0x0(0)/0x0(0)/0x0(0)
    Last retransmission scan length is 1, maximum is 2
    Last retransmission scan time is 0 msec, maximum is 0 msec
 Neighbor 172.16.5.5
    In the area 2 via interface ATM3/0
    Neighbor: interface-id 13, link-local address FE80::205:5FFF:FED3:6006
    Neighbor priority is 1, State is FULL, 6 state changes
    Options is 0x63F7D249
    Dead timer due in 00:00:38
    Neighbor is up for 00:10:01
    Index 1/1/3, retransmission queue length 0, number of retransmission 0
    First 0x0(0)/0x0(0)/0x0(0) Next 0x0(0)/0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Additional References for BFD IPv6 Encapsulation Support

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	IPv6 RFCs

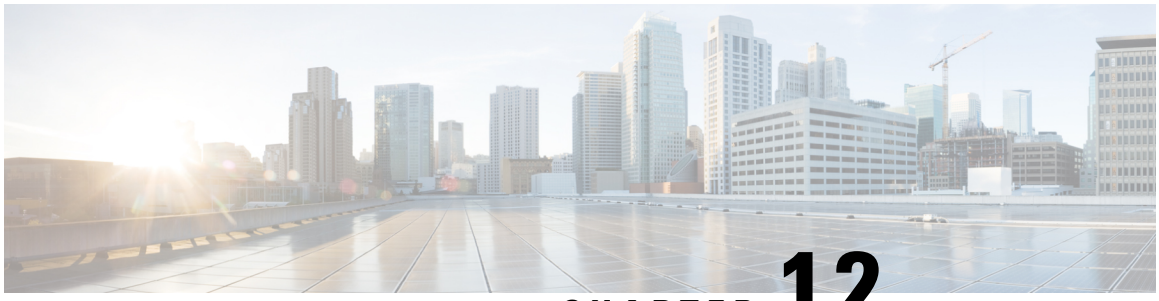
Feature Information for BFD IPv6 Encapsulation Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Table 23: Feature Information for BFD IPv6 Encapsulation Support

Feature Name	Releases	Feature Information
BFD IPv6 Encapsulation Support	Cisco IOS XE Everest 16.6.2	BFDv6 encapsulations are described within a session information structure. These session information structures are defined by BFDv6 for the protocols supported. BFDv6 uses information from the session information structures to determine the correct encapsulation for BFDv6 packets on that session. In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.



CHAPTER 12

Configuring BGP

Restrictions for BGP

The BGP hold time must always be configured higher than the Graceful Restart hold time on a device, even with Graceful Restart disabled. A peer device with an unsupported hold time can establish a session with a device through an open message, but once Graceful Restart is enabled the session will flap.

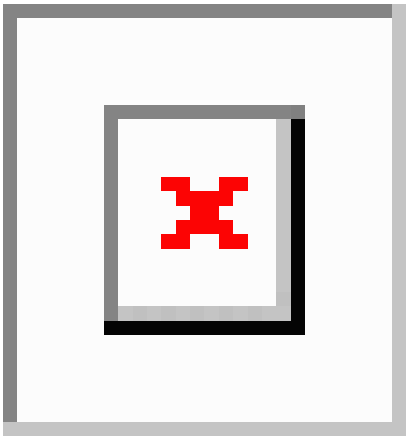
Information About BGP

The Border Gateway Protocol (BGP) is an exterior gateway protocol used to set up an interdomain routing system that guarantees the loop-free exchange of routing information between autonomous systems. Autonomous systems are made up of routers that operate under the same administration and that run Interior Gateway Protocols (IGPs), such as RIP or OSPF, within their boundaries and that interconnect by using an Exterior Gateway Protocol (EGP). BGP Version 4 is the standard EGP for interdomain routing in the Internet. The protocol is defined in RFCs 1163, 1267, and 1771.

BGP Network Topology

Routers that belong to the same autonomous system (AS) and that exchange BGP updates run internal BGP (IBGP), and routers that belong to different autonomous systems and that exchange BGP updates run external BGP (EBGP). Most configuration commands are the same for configuring EBGP and IBGP. The difference is that the routing updates are exchanged either between autonomous systems (EBGP) or within an AS (IBGP). The figure given below shows a network that is running both EBGP and IBGP.

Figure 6: EBGP, IBGP, and Multiple Autonomous Systems



Before exchanging information with an external AS, BGP ensures that networks within the AS can be reached by defining internal BGP peering among routers within the AS and by redistributing BGP routing information to IGPs that run within the AS, such as IGRP and OSPF.

Routers that run a BGP routing process are often referred to as BGP speakers. BGP uses the Transmission Control Protocol (TCP) as its transport protocol (specifically port 179). Two BGP speakers that have a TCP connection to each other for exchanging routing information are known as peers or neighbors. In the above figure, Routers A and B are BGP peers, as are Routers B and C and Routers C and D. The routing information is a series of AS numbers that describe the full path to the destination network. BGP uses this information to construct a loop-free map of autonomous systems.

The network has these characteristics:

- Routers A and B are running EBGP, and Routers B and C are running IBGP. Note that the EBGP peers are directly connected and that the IBGP peers are not. As long as there is an IGP running that allows the two neighbors to reach one another, IBGP peers do not have to be directly connected.
- All BGP speakers within an AS must establish a peer relationship with each other. That is, the BGP speakers within an AS must be fully meshed logically. BGP4 provides two techniques that reduce the requirement for a logical full mesh: confederations and route reflectors.
- AS 200 is a transit AS for AS 100 and AS 300—that is, AS 200 is used to transfer packets between AS 100 and AS 300.

BGP peers initially exchange their full BGP routing tables and then send only incremental updates. BGP peers also exchange keepalive messages (to ensure that the connection is up) and notification messages (in response to errors or special conditions).

In BGP, each route consists of a network number, a list of autonomous systems that information has passed through (the autonomous system path), and a list of other path attributes. The primary function of a BGP system is to exchange network reachability information, including information about the list of AS paths, with other BGP systems. This information can be used to determine AS connectivity, to prune routing loops, and to enforce AS-level policy decisions.

A router or Device running Cisco IOS does not select or use an IBGP route unless it has a route available to the next-hop router and it has received synchronization from an IGP (unless IGP synchronization is disabled). When multiple routes are available, BGP bases its path selection on attribute values. See the “Configuring BGP Decision Attributes” section for information about BGP attributes.

BGP Version 4 supports classless interdomain routing (CIDR) so you can reduce the size of your routing tables by creating aggregate routes, resulting in supernets. CIDR eliminates the concept of network classes within BGP and supports the advertising of IP prefixes.

Nonstop Forwarding Awareness

The BGP NSF Awareness feature is supported for IPv4 in the . To enable this feature with BGP routing, you need to enable Graceful Restart. When the neighboring router is NSF-capable, and this feature is enabled, the Layer 3 Device continues to forward packets from the neighboring router during the interval between the primary Route Processor (RP) in a router failing and the backup RP taking over, or while the primary RP is manually reloaded for a nondisruptive software upgrade.

Information About BGP Routing

To enable BGP routing, you establish a BGP routing process and define the local network. Because BGP must completely recognize the relationships with its neighbors, you must also specify a BGP neighbor.

BGP supports two kinds of neighbors: internal and external. Internal neighbors are in the same AS; external neighbors are in different autonomous systems. External neighbors are usually adjacent to each other and share a subnet, but internal neighbors can be anywhere in the same AS.

The switch supports the use of private AS numbers, usually assigned by service providers and given to systems whose routes are not advertised to external neighbors. The private AS numbers are from 64512 to 65535. You can configure external neighbors to remove private AS numbers from the AS path by using the **neighbor remove-private-as** router configuration command. Then when an update is passed to an external neighbor, if the AS path includes private AS numbers, these numbers are dropped.

If your AS will be passing traffic through it from another AS to a third AS, it is important to be consistent about the routes it advertises. If BGP advertised a route before all routers in the network had learned about the route through the IGP, the AS might receive traffic that some routers could not yet route. To prevent this from happening, BGP must wait until the IGP has propagated information across the AS so that BGP is synchronized with the IGP. Synchronization is enabled by default. If your AS does not pass traffic from one AS to another AS, or if all routers in your autonomous systems are running BGP, you can disable synchronization, which allows your network to carry fewer routes in the IGP and allows BGP to converge more quickly.

Routing Policy Changes

Routing policies for a peer include all the configurations that might affect inbound or outbound routing table updates. When you have defined two routers as BGP neighbors, they form a BGP connection and exchange routing information. If you later change a BGP filter, weight, distance, version, or timer, or make a similar configuration change, you must reset the BGP sessions so that the configuration changes take effect.

There are two types of reset, hard reset and soft reset. Cisco IOS Releases 12.1 and later support a soft reset without any prior configuration. To use a soft reset without preconfiguration, both BGP peers must support the soft route refresh capability, which is advertised in the OPEN message sent when the peers establish a TCP session. A soft reset allows the dynamic exchange of route refresh requests and routing information between BGP routers and the subsequent re-advertisement of the respective outbound routing table.

- When soft reset generates inbound updates from a neighbor, it is called dynamic inbound soft reset.
- When soft reset sends a set of updates to a neighbor, it is called outbound soft reset.

A soft inbound reset causes the new inbound policy to take effect. A soft outbound reset causes the new local outbound policy to take effect without resetting the BGP session. As a new set of updates is sent during outbound policy reset, a new inbound policy can also take effect.

The table given below lists the advantages and disadvantages hard reset and soft reset.

Table 24: Advantages and Disadvantages of Hard and Soft Resets

Type of Reset	Advantages	Disadvantages
Hard reset	No memory overhead	The prefixes in the BGP, IP, and FIB t provided by the neighbor are lost. Not recommended.
Outbound soft reset	No configuration, no storing of routing table updates	Does not reset inbound routing table u
Dynamic inbound soft reset	Does not clear the BGP session and cache Does not require storing of routing table updates and has no memory overhead	Both BGP routers must support the ro capability (in Cisco IOS Release 12.1

BGP Decision Attributes

When a BGP speaker receives updates from multiple autonomous systems that describe different paths to the same destination, it must choose the single best path for reaching that destination. When chosen, the selected path is entered into the BGP routing table and propagated to its neighbors. The decision is based on the value of attributes that the update contains and other BGP-configurable factors.

When a BGP peer learns two EBGP paths for a prefix from a neighboring AS, it chooses the best path and inserts that path in the IP routing table. If BGP multipath support is enabled and the EBGP paths are learned from the same neighboring autonomous systems, instead of a single best path, multiple paths are installed in the IP routing table. Then, during packet switching, per-packet or per-destination load-balancing is performed among the multiple paths. The **maximum-paths** router configuration command controls the number of paths allowed.

These factors summarize the order in which BGP evaluates the attributes for choosing the best path:

1. If the path specifies a next hop that is inaccessible, drop the update. The BGP next-hop attribute, automatically determined by the software, is the IP address of the next hop that is going to be used to reach a destination. For EBGP, this is usually the IP address of the neighbor specified by the **neighbor remote-as router** configuration command. You can disable next-hop processing by using route maps or the **neighbor next-hop-self** router configuration command.
2. Prefer the path with the largest weight (a Cisco proprietary parameter). The weight attribute is local to the router and not propagated in routing updates. By default, the weight attribute is 32768 for paths that the router originates and zero for other paths. Routes with the largest weight are preferred. You can use access lists, route maps, or the **neighbor weight** router configuration command to set weights.
3. Prefer the route with the highest local preference. Local preference is part of the routing update and exchanged among routers in the same AS. The default value of the local preference attribute is 100. You can set local preference by using the **bgp default local-preference** router configuration command or by using a route map.
4. Prefer the route that was originated by BGP running on the local router.

5. Prefer the route with the shortest AS path.
6. Prefer the route with the lowest origin type. An interior route or IGP is lower than a route learned by EGP, and an EGP-learned route is lower than one of unknown origin or learned in another way.
7. Prefer the route with the lowest multi -exit discriminator (MED) metric attribute if the neighboring AS is the same for all routes considered. You can configure the MED by using route maps or by using the **default-metric** router configuration command. When an update is sent to an IBGP peer, the MED is included.
8. Prefer the external (EBGP) path over the internal (IBGP) path.
9. Prefer the route that can be reached through the closest IGP neighbor (the lowest IGP metric). This means that the router will prefer the shortest internal path within the AS to reach the destination (the shortest path to the BGP next-hop).
10. If the following conditions are all true, insert the route for this path into the IP routing table:

    Both the best route and this route are external.

    Both the best route and this route are from the same neighboring autonomous system.

    Maximum-paths is enabled.
11. If multipath is not enabled, prefer the route with the lowest IP address value for the BGP router ID. The router ID is usually the highest IP address on the router or the loopback (virtual) address, but might be implementation-specific.

Route Maps

Within BGP, route maps can be used to control and to modify routing information and to define the conditions by which routes are redistributed between routing domains. Each route map has a name that identifies the route map (*map tag*) and an optional sequence number.

BGP Filtering

You can filter BGP advertisements by using AS-path filters, such as the **as-path access-list** global configuration command and the **neighbor filter-list** router configuration command. You can also use access lists with the **neighbor distribute-list** router configuration command. Distribute-list filters are applied to network numbers. See the “Controlling Advertising and Processing in Routing Updates” section for information about the **distribute-list** command.

You can use route maps on a per-neighbor basis to filter updates and to modify various attributes. A route map can be applied to either inbound or outbound updates. Only the routes that pass the route map are sent or accepted in updates. On both inbound and outbound updates, matching is supported based on AS path, community, and network numbers. Autonomous system path matching requires the **match as-path access-list** route-map command, community based matching requires the **match community-list** route-map command, and network-based matching requires the **ip access-list** global configuration command.

Prefix List for BGP Filtering

You can use prefix lists as an alternative to access lists in many BGP route filtering commands, including the **neighbor distribute-list** router configuration command. The advantages of using prefix lists include

performance improvements in loading and lookup of large lists, incremental update support, easier CLI configuration, and greater flexibility.

Filtering by a prefix list involves matching the prefixes of routes with those listed in the prefix list, as when matching access lists. When there is a match, the route is used. Whether a prefix is permitted or denied is based upon these rules:

- An empty prefix list permits all prefixes.
- An implicit deny is assumed if a given prefix does not match any entries in a prefix list.
- When multiple entries of a prefix list match a given prefix, the sequence number of a prefix list entry identifies the entry with the lowest sequence number.

By default, sequence numbers are generated automatically and incremented in units of five. If you disable the automatic generation of sequence numbers, you must specify the sequence number for each entry. You can specify sequence values in any increment. If you specify increments of one, you cannot insert additional entries into the list; if you choose very large increments, you might run out of values.

BGP Community Filtering

One way that BGP controls the distribution of routing information based on the value of the COMMUNITIES attribute. The attribute is a way to groups destinations into communities and to apply routing decisions based on the communities. This method simplifies configuration of a BGP speaker to control distribution of routing information.

A community is a group of destinations that share some common attribute. Each destination can belong to multiple communities. AS administrators can define to which communities a destination belongs. By default, all destinations belong to the general Internet community. The community is identified by the COMMUNITIES attribute, an optional, transitive, global attribute in the numerical range from 1 to 4294967200. These are some predefined, well-known communities:

- **internet**—Advertise this route to the Internet community. All routers belong to it.
- **no-export**—Do not advertise this route to EBGP peers.
- **no-advertise**—Do not advertise this route to any peer (internal or external).
- **local-as**—Do not advertise this route to peers outside the local autonomous system.

Based on the community, you can control which routing information to accept, prefer, or distribute to other neighbors. A BGP speaker can set, append, or modify the community of a route when learning, advertising, or redistributing routes. When routes are aggregated, the resulting aggregate has a COMMUNITIES attribute that contains all communities from all the initial routes.

You can use community lists to create groups of communities to use in a match clause of a route map. As with an access list, a series of community lists can be created. Statements are checked until a match is found. As soon as one statement is satisfied, the test is concluded.

BGP Neighbors and Peer Groups

Often many BGP neighbors are configured with the same update policies (that is, the same outbound route maps, distribute lists, filter lists, update source, and so on). Neighbors with the same update policies can be

grouped into peer groups to simplify configuration and to make updating more efficient. When you have configured many peers, we recommend this approach.

To configure a BGP peer group, you create the peer group, assign options to the peer group, and add neighbors as peer group members. You configure the peer group by using the **neighbor** router configuration commands. By default, peer group members inherit all the configuration options of the peer group, including the remote-as (if configured), version, update-source, out-route-map, out-filter-list, out-dist-list, minimum-advertisement-interval, and next-hop-self. All peer group members also inherit changes made to the peer group. Members can also be configured to override the options that do not affect outbound updates.

Aggregate Routes

Classless interdomain routing (CIDR) enables you to create aggregate routes (or supernets) to minimize the size of routing tables. You can configure aggregate routes in BGP either by redistributing an aggregate route into BGP or by creating an aggregate entry in the BGP routing table. An aggregate address is added to the BGP table when there is at least one more specific entry in the BGP table.

Routing Domain Confederations

One way to reduce the IBGP mesh is to divide an autonomous system into multiple subautonomous systems and to group them into a single confederation that appears as a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections to other autonomous systems in the same confederation. Even though the peers in different autonomous systems have EBGP sessions, they exchange routing information as if they were IBGP peers. Specifically, the next hop, MED, and local preference information is preserved. You can then use a single IGP for all of the autonomous systems.

BGP Route Reflectors

BGP requires that all of the IBGP speakers be fully meshed. When a router receives a route from an external neighbor, it must advertise it to all internal neighbors. To prevent a routing information loop, all IBPG speakers must be connected. The internal neighbors do not send routes learned from internal neighbors to other internal neighbors.

With route reflectors, all IBGP speakers need not be fully meshed because another method is used to pass learned routes to neighbors. When you configure an internal BGP peer to be a route reflector, it is responsible for passing IBGP learned routes to a set of IBGP neighbors. The internal peers of the route reflector are divided into two groups: client peers and nonclient peers (all the other routers in the autonomous system). A route reflector reflects routes between these two groups. The route reflector and its client peers form a cluster. The nonclient peers must be fully meshed with each other, but the client peers need not be fully meshed. The clients in the cluster do not communicate with IBGP speakers outside their cluster.

When the route reflector receives an advertised route, it takes one of these actions, depending on the neighbor:

- A route from an external BGP speaker is advertised to all clients and nonclient peers.
- A route from a nonclient peer is advertised to all clients.
- A route from a client is advertised to all clients and nonclient peers. Hence, the clients need not be fully meshed.

Usually a cluster of clients have a single route reflector, and the cluster is identified by the route reflector router ID. To increase redundancy and to avoid a single point of failure, a cluster might have more than one

route reflector. In this case, all route reflectors in the cluster must be configured with the same 4-byte cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. All the route reflectors serving a cluster should be fully meshed and should have identical sets of client and nonclient peers.

Route Dampening

Route flap dampening is a BGP feature designed to minimize the propagation of flapping routes across an internetwork. A route is considered to be flapping when it is repeatedly available, then unavailable, then available, then unavailable, and so on. When route dampening is enabled, a numeric penalty value is assigned to a route when it flaps. When a route's accumulated penalties reach a configurable limit, BGP suppresses advertisements of the route, even if the route is running. The reuse limit is a configurable value that is compared with the penalty. If the penalty is less than the reuse limit, a suppressed route that is up is advertised again.

Dampening is not applied to routes that are learned by IBGP. This policy prevents the IBGP peers from having a higher penalty for routes external to the AS.

More BGP Information

For detailed descriptions of BGP configuration, see the "Configuring BGP" chapter in the "IP Routing Protocols" part of the *Cisco IOS IP Configuration Guide, Release 12.4*. For details about specific commands, see the *Cisco IOS IP Command Reference, Volume 2 of 3: Routing Protocols, Release 12.4*.

How to Configure BGP

Default BGP Configuration

The table given below shows the basic default BGP configuration.

Table 25: Default BGP Configuration

Feature	Default Setting
Aggregate address	Disabled: None defined.
AS path access list	None defined.
Auto summary	Disabled.
Best path	• The router considers *as-path* in choosing a route and does not compare simi from external BGP peers. • Compare router ID: Disabled.
BGP community list	• Number: None defined. When you permit a value for the community numbe defaults to an implicit deny for everything else that has not been permitted. • Format: Cisco default format (32-bit number).

Feature	Default Setting
BGP confederation identifier/peers	• Identifier: None configured. • Peers: None identified.
BGP Fast external fallover	Enabled.
BGP local preference	100. The range is 0 to 4294967295 with the higher value preferred.
BGP network	None specified; no backdoor route advertised.
BGP route dampening	Disabled by default. When enabled: • Half-life is 15 minutes. • Re-use is 750 (10-second increments). • Suppress is 2000 (10-second increments). • Max-suppress-time is 4 times half-life; 60 minutes.
BGP router ID	The IP address of a loopback interface if one is configured or the highest IP addr for a physical interface on the router.
Default information originate (protocol or network redistribution)	Disabled.
Default metric	Built-in, automatic metric translations.
Distance	• External route administrative distance: 20 (acceptable values are from 1 • Internal route administrative distance: 200 (acceptable values are from 1 • Local route administrative distance: 200 (acceptable values are from 1 t
Distribute list	• In (filter networks received in updates): Disabled. • Out (suppress networks from being advertised in updates): Disabled.
Internal route redistribution	Disabled.
IP prefix list	None defined.
Multi exit discriminator (MED)	• Always compare: Disabled. Does not compare MEDs for paths from nei different autonomous systems. • Best path compare: Disabled. • MED missing as worst path: Disabled. • Deterministic MED comparison is disabled.

Feature	Default Setting
Neighbor	• Advertisement interval: 30 seconds for external peers; 5 seconds for interna • Change logging: Enabled. • Conditional advertisement: Disabled. • Default originate: No default route is sent to the neighbor. • Description: None. • Distribute list: None defined. • External BGP multihop: Only directly connected neighbors are allowed. • Filter list: None used. • Maximum number of prefixes received: No limit. • Next hop (router as next hop for BGP neighbor): Disabled. • Password: Disabled. • Peer group: None defined; no members assigned. • Prefix list: None specified. • Remote AS (add entry to neighbor BGP table): No peers defined. • Private AS number removal: Disabled. • Route maps: None applied to a peer. • Send community attributes: None sent to neighbors. • Shutdown or soft reconfiguration: Not enabled. • Timers: keepalive: 60 seconds; holdtime: 180 seconds. • Update source: Best local address. • Version: BGP Version 4. • Weight: Routes learned through BGP peer: 0; routes sourced by the local rout
NSF[1] Awareness	Disabled[2]. If enabled, allows Layer 3 switches to continue forwarding packets fr neighboring NSF-capable router during hardware or software changes.
Route reflector	None configured.
Synchronization (BGP and IGP)	Disabled.
Table map update	Disabled.
Timers	Keepalive: 60 seconds; holdtime: 180 seconds.

[1] Nonstop Forwarding
[2] NSF Awareness can be enabled for IPv4 on switches with the license by enabling Graceful Restart.

Enabling BGP Routing

Before you begin

Note To enable BGP, the switch or active switch must be running the

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 45000`	Enables a BGP routing process, assign it an AS number, and enter router configuration mode. The AS number can be from 1 to 65535, with 64512 to 65535 designated as private autonomous numbers.
Step 4	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Device(config-router)# network 10.108.0.0`	Configures a network as local to this AS, and enter it in the BGP table.
Step 5	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *number* **Example:** `Device(config-router)# neighbor 10.108.1.2 remote-as 65200`	Adds an entry to the BGP neighbor table specifying that the neighbor identified by the IP address belongs to the specified AS. For EBGP, neighbors are usually directly connected, and the IP address is the address of the interface at the other end of the connection. For IBGP, the IP address can be the address of any of the router interfaces.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **remove-private-as** **Example:** `Device(config-router)# neighbor 172.16.2.33 remove-private-as`	(Optional) Removes private AS numbers from the AS-path in outbound routing updates.

	Command or Action	Purpose
Step 7	**synchronization** **Example:** `Device(config-router)# synchronization`	(Optional) Enables synchronization between BGP and an IGP.
Step 8	**auto-summary** **Example:** `Device(config-router)# auto-summary`	(Optional) Enables automatic network summarization. When a subnet is redistributed from an IGP into BGP, only the network route is inserted into the BGP table.
Step 9	**bgp graceful-restart** **Example:** `Device(config-router)# bgp graceful-start`	(Optional) Enables NSF awareness on switch. By default, NSF awareness is disabled.
Step 10	**end** **Example:** `Device(config-router)#end`	Returns to privileged EXEC mode.
Step 11	**show ip bgp network** *network-number* **Example:** `Device# show ip bgp network 10.108.0.0`	Verifies the configuration.
Step 12	**show ip bgp neighbor** **Example:** `Device# show ip bgp neighbor`	Verifies that NSF awareness (Graceful Restart) is enabled on the neighbor. If NSF awareness is enabled on the switch and the neighbor, this message appears: *Graceful Restart Capability: advertised and received* If NSF awareness is enabled on the switch, but not on the neighbor, this message appears: *Graceful Restart Capability: advertised*
Step 13	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Managing Routing Policy Changes

To learn if a BGP peer supports the route refresh capability and to reset the BGP session:

Procedure

	Command or Action	Purpose
Step 1	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Displays whether a neighbor supports the route refresh capability. When supported, this message appears for the router: *Received route refresh capability from peer.*
Step 2	**clear ip bgp** {***** | *address* | *peer-group-name*} **Example:** `Device# clear ip bgp *`	Resets the routing table on the specified connection. • Enter an asterisk (*) to specify that all connections be reset. • Enter an IP address to specify the connection to be reset. • Enter a peer group name to reset the peer group.
Step 3	**clear ip bgp** {***** | *address* | *peer-group-name*} **soft out** **Example:** `Device# clear ip bgp * soft out`	(Optional) Performs an outbound soft reset to reset the inbound routing table on the specified connection. Use this command if route refresh is supported. • Enter an asterisk (*) to specify that all connections be reset. • Enter an IP address to specify the connection to be reset. • Enter a peer group name to reset the peer group.
Step 4	**show ip bgp** **Example:** `Device# show ip bgp`	Verifies the reset by checking information about the routing table and about BGP neighbors.
Step 5	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Verifies the reset by checking information about the routing table and about BGP neighbors.

Configuring BGP Decision Attributes

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters the global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 2	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 4500`	Enables a BGP routing process, assign it an AS number, and enter router configuration mode.
Step 3	**bgp best-path as-path ignore** **Example:** `Device(config-router)# bgp bestpath as-path ignore`	(Optional) Configures the router to ignore AS path length in selecting a route.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **next-hop-self** **Example:** `Device(config-router)# neighbor 10.108.1.1 next-hop-self`	(Optional) Disables next-hop processing on BGP updates to a neighbor by entering a specific IP address to be used instead of the next-hop address.
Step 5	**neighbor** {*ip-address* | *peer-group-name*} **weight** *weight* **Example:** `Device(config-router)# neighbor 172.16.12.1 weight 50`	(Optional) Assign a weight to a neighbor connection. Acceptable values are from 0 to 65535; the largest weight is the preferred route. Routes learned through another BGP peer have a default weight of 0; routes sourced by the local router have a default weight of 32768.
Step 6	**default-metric** *number* **Example:** `Device(config-router)# default-metric 300`	(Optional) Sets a MED metric to set preferred paths to external neighbors. All routes without a MED will also be set to this value. The range is 1 to 4294967295. The lowest value is the most desirable.
Step 7	**bgp bestpath med missing-as-worst** **Example:** `Device(config-router)# bgp bestpath med missing-as-worst`	(Optional) Configures the switch to consider a missing MED as having a value of infinity, making the path without a MED value the least desirable path.
Step 8	**bgp always-compare med** **Example:** `Device(config-router)# bgp always-compare-med`	(Optional) Configures the switch to compare MEDs for paths from neighbors in different autonomous systems. By default, MED comparison is only done among paths in the same AS.
Step 9	**bgp bestpath med confed** **Example:** `Device(config-router)# bgp bestpath med confed`	(Optional) Configures the switch to consider the MED in choosing a path from among those advertised by different subautonomous systems within a confederation.

	Command or Action	Purpose
Step 10	**bgp deterministic med** **Example:** `Device(config-router)# bgp deterministic med`	(Optional) Configures the switch to consider the MED variable when choosing among routes advertised by different peers in the same AS.
Step 11	**bgp default local-preference** *value* **Example:** `Device(config-router)# bgp default local-preference 200`	(Optional) Change the default local preference value. The range is 0 to 4294967295; the default value is 100. The highest local preference value is preferred.
Step 12	**maximum-paths** *number* **Example:** `Device(config-router)# maximum-paths 8`	(Optional) Configures the number of paths to be added to the IP routing table. The default is to only enter the best path in the routing table. The range is from 1 to 16. Having multiple paths allows load-balancing among the paths. (Although the switch software allows a maximum of 32 equal-cost routes, the switch hardware will never use more than 16 paths per route.)
Step 13	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 14	**show ip bgp** **Example:** `Device# show ip bgp`	Verifies the reset by checking information about the routing table and about BGP neighbors.
Step 15	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Verifies the reset by checking information about the routing table and about BGP neighbors.
Step 16	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Filtering with Route Maps

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map set-peer-address permit 10`	Creates a route map, and enter route-map configuration mode.
Step 3	**set ip next-hop** *ip-address* [*...ip-address*] [**peer-address**] **Example:** `Device(config)# set ip next-hop 10.1.1.3`	(Optional) Sets a route map to disable next-hop processing • In an inbound route map, set the next hop of matching routes to be the neighbor peering address, overriding third-party next hops. • In an outbound route map of a BGP peer, set the next hop to the peering address of the local router, disabling the next-hop calculation.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show route-map** [*map-name*] **Example:** `Device# show route-map`	Displays all route maps configured or only the one specified to verify configuration.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Filtering by Neighbor

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 109`	Enables a BGP routing process, assign it an AS number, and enter router configuration mode.
Step 3	**neighbor** {*ip-address* | *peer-group name*} **distribute-list** {**access-list-number** | *name*} {**in** | **out**} **Example:** `Device(config-router)# neighbor 172.16.4.1 distribute-list 39 in`	(Optional) Filters BGP routing updates to or from neighbors as specified in an access list. **Note** You can also use the **neighbor prefix-list** router configuration command to filter updates, but you cannot use both commands to configure the same BGP peer.
Step 4	**neighbor** {*ip-address* | *peer-group name*} **route-map** *map-tag* {**in** | **out**} **Example:** `Device(config-router)# neighbor 172.16.70.24 route-map internal-map in`	(Optional) Applies a route map to filter an incoming or outgoing route.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Verifies the configuration.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Filtering by Access Lists and Neighbors

Another method of filtering is to specify an access list filter on both incoming and outbound updates, based on the BGP autonomous system paths. Each filter is an access list based on regular expressions. To use this method, define an autonomous system path access list, and apply it to updates to and from particular neighbors.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip as-path access-list** *access-list-number* {**permit** | **deny**} *as-regular-expressions* **Example:** `Device(config)# ip as-path access-list 1 deny _65535_`	Defines a BGP-related access list.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 110`	Enters BGP router configuration mode.
Step 4	**neighbor** {*ip-address* | *peer-group name*} **filter-list** {*access-list-number* | *name*} {**in** | **out** | **weight** *weight*} **Example:** `Device(config-router)# neighbor 172.16.1.1 filter-list 1 out`	Establishes a BGP filter based on an access list.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ip bgp neighbors** [**paths** *regular-expression*] **Example:** `Device# show ip bgp neighbors`	Verifies the configuration.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Prefix Lists for BGP Filtering

You do not need to specify a sequence number when removing a configuration entry. **Show** commands include the sequence numbers in their output.

Before using a prefix list in a command, you must set up the prefix list.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip prefix-list** *list-name* [**seq** *seq-value*] **deny** | **permit** *network/len* [**ge** *ge-value*] [**le** *le-value*] **Example:** `Device(config)# ip prefix-list BLUE permit 172.16.1.0/24`	Creates a prefix list with an optional sequence number to **deny** or **permit** access for matching conditions. You must enter at least one **permit** or **deny** clause. • *network/len* is the network number and length (in bits) of the network mask. • (Optional) **ge** and **le** values specify the range of the prefix length to be matched.The specified *ge-value* and *le-value* must satisfy this condition: *len* < *ge-value* < *le-value* < 32
Step 3	**ip prefix-list** *list-name* **seq** *seq-value* **deny** | **permit** *network/len* [**ge** *ge-value*] [**le** *le-value*] **Example:** `Device(config)# ip prefix-list BLUE seq 10 permit 172.24.1.0/24`	(Optional) Adds an entry to a prefix list, and assign a sequence number to the entry.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show ip prefix list** [**detail** | **summary**] *name* [*network/len*] [**seq** *seq-num*] [**longer**] [**first-match**] **Example:** `Device# show ip prefix list summary test`	Verifies the configuration by displaying information about a prefix list or prefix list entries.
Step 6	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Configuring BGP Community Filtering

By default, no COMMUNITIES attribute is sent to a neighbor. You can specify that the COMMUNITIES attribute be sent to the neighbor at an IP address by using the **neighbor send-community** router configuration command.

SUMMARY STEPS

1. **configure terminal**
2. **ip community-list** *community-list-number* {**permit** | **deny**} *community-number*
3. **router bgp** *autonomous-system*
4. **neighbor** {*ip-address* | *peer-group name*} **send-community**
5. **set comm-list** *list-num* **delete**
6. **exit**
7. **ip bgp-community new-format**
8. **end**
9. **show ip bgp community**
10. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip community-list** *community-list-number* {**permit**	**deny**} *community-number* **Example:** `Device(config)# ip community-list 1 permit 50000:10`	Creates a community list, and assigns it a number. • The *community-list-number* is an integer from 1 to 99 that identifies one or more permit or deny groups of communities. • The *community-number* is the number configured by a **set community** route-map configuration command.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 108`	Enters BGP router configuration mode.

	Command or Action	Purpose
Step 4	**neighbor** {*ip-address* | *peer-group name*} **send-community** **Example:** `Device(config-router)# neighbor 172.16.70.23 send-community`	Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.
Step 5	**set comm-list** *list-num* **delete** **Example:** `Device(config-router)# set comm-list 500 delete`	(Optional) Removes communities from the community attribute of an inbound or outbound update that match a standard or extended community list specified by a route map.
Step 6	**exit** **Example:** `Device(config-router)# end`	Returns to global configuration mode.
Step 7	**ip bgp-community new-format** **Example:** `Device(config)# ip bgp-community new format`	(Optional) Displays and parses BGP communities in the format AA:NN. A BGP community is displayed in a two-part format 2 bytes long. The Cisco default community format is in the format NNAA. In the most recent RFC for BGP, a community takes the form AA:NN, where the first part is the AS number and the second part is a 2-byte number.
Step 8	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 9	**show ip bgp community** **Example:** `Device# show ip bgp community`	Verifies the configuration.
Step 10	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Neighbors and Peer Groups

To assign configuration options to an individual neighbor, specify any of these router configuration commands by using the neighbor IP address. To assign the options to a peer group, specify any of the commands by using

the peer group name. You can disable a BGP peer or peer group without removing all the configuration information by using the **neighbor shutdown** router configuration command.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router bgp** *autonomous-system*	Enters BGP router configuration mode.
Step 3	**neighbor** *peer-group-name* **peer-group**	Creates a BGP peer group.
Step 4	**neighbor** *ip-address* **peer-group** *peer-group-name*	Makes a BGP neighbor a member of the peer group.
Step 5	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *number*	Specifies a BGP neighbor. If a peer group is not configured with a **remote-as** *number*, use this command to create peer groups containing EBGP neighbors. The range is 1 to 65535.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **description** *text*	(Optional) Associates a description with a neighbor.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **default-originate** [**route-map** *map-name*]	(Optional) Allows a BGP speaker (the local router) to send the default route 0.0.0.0 to a neighbor for use as a default route.
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **send-community**	(Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.
Step 9	**neighbor** {*ip-address* | *peer-group-name*} **update-source** *interface*	(Optional) Allows internal BGP sessions to use any operational interface for TCP connections.
Step 10	**neighbor** {*ip-address* | *peer-group-name*} **ebgp-multihop**	(Optional) Allows BGP sessions, even when the neighbor is not on a directly connected segment. The multihop session is not established if the only route to the multihop peer's address is the default route (0.0.0.0).
Step 11	**neighbor** {*ip-address* | *peer-group-name*} **local-as** *number*	(Optional) Specifies an AS number to use as the local AS. The range is 1 to 65535.
Step 12	**neighbor** {*ip-address* | *peer-group-name*} **advertisement-interval** *seconds*	(Optional) Sets the minimum interval between sending BGP routing updates.
Step 13	**neighbor** {*ip-address* | *peer-group-name*} **maximum-prefix** *maximum* [*threshold*]	(Optional) Controls how many prefixes can be received from a neighbor. The range is 1 to 4294967295. The *threshold* (optional) is the percentage of maximum at which a warning message is generated. The default is 75 percent.

	Command or Action	Purpose
Step 14	**neighbor** {*ip-address* | *peer-group-name*} **next-hop-self**	(Optional) Disables next-hop processing on the BGP updates to a neighbor.
Step 15	neighbor {*ip-address* | *peer-group-name*} password *string*	(Optional) Sets MD5 authentication on a TCP connection to a BGP peer. The same password must be configured on both BGP peers, or the connection between them is not made.
Step 16	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name* {**in** | **out**}	(Optional) Applies a route map to incoming or outgoing routes.
Step 17	**neighbor** {*ip-address* | *peer-group-name*} **send-community**	(Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.
Step 18	**neighbor** {*ip-address* | *peer-group-name*} **timers** *keepalive holdtime*	(Optional) Sets timers for the neighbor or peer group. • The *keepalive* interval is the time within which keepalive messages are sent to peers. The range is 1 to 4294967295 seconds; the default is 60. • The *holdtime* is the interval after which a peer is declared inactive after not receiving a keepalive message from it. The range is 1 to 4294967295 seconds; the default is 180.
Step 19	**neighbor** {*ip-address* | *peer-group-name*} **weight** *weight*	(Optional) Specifies a weight for all routes from a neighbor.
Step 20	**neighbor** {*ip-address* | *peer-group-name*} **distribute-list** {*access-list-number* | *name*} {**in** | **out**}	(Optional) Filter BGP routing updates to or from neighbors, as specified in an access list.
Step 21	**neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number* {**in** | **out** | **weight** *weight*}	(Optional) Establish a BGP filter.
Step 22	**neighbor** {*ip-address* | *peer-group-name*} **version** *value*	(Optional) Specifies the BGP version to use when communicating with a neighbor.
Step 23	**neighbor** {*ip-address* | *peer-group-name*} **soft-reconfiguration inbound**	(Optional) Configures the software to start storing received updates.
Step 24	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 25	**show ip bgp neighbors**	Verifies the configuration.
Step 26	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Configuring Aggregate Addresses in a Routing Table

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 106`	Enters BGP router configuration mode.
Step 3	**aggregate-address** *address mask* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0`	Creates an aggregate entry in the BGP routing table. The aggregate route is advertised as coming from the AS, and the atomic aggregate attribute is set to indicate that information might be missing.
Step 4	**aggregate-address** *address mask* **as-set** **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 as-set`	(Optional) Generates AS set path information. This command creates an aggregate entry following the same rules as the previous command, but the advertised path will be an AS_SET consisting of all elements contained in all paths. Do not use this keyword when aggregating many paths because this route must be continually withdrawn and updated.
Step 5	**aggregate-address** *address-mask* **summary-only** **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 summary-only`	(Optional) Advertises summary addresses only.
Step 6	**aggregate-address** *address mask* **suppress-map** *map-name* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 suppress-map map1`	(Optional) Suppresses selected, more specific routes.

	Command or Action	Purpose
Step 7	**aggregate-address** *address mask* **advertise-map** *map-name* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 advertise-map map2`	(Optional) Generates an aggregate based on conditions specified by the route map.
Step 8	**aggregate-address** *address mask* **attribute-map** *map-name* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 attribute-map map3`	(Optional) Generates an aggregate with attributes specified in the route map.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 10	**show ip bgp neighbors** [**advertised-routes**] **Example:** `Device# show ip bgp neighbors`	Verifies the configuration.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Routing Domain Confederations

You must specify a confederation identifier that acts as the autonomous system number for the group of autonomous systems.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router bgp** *autonomous-system* **Example:**	Enters BGP router configuration mode.

	Command or Action	Purpose
	`Device(config)# router bgp 100`	
Step 3	**bgp confederation identifier** *autonomous-system* **Example:** `Device(config)# bgp confederation identifier 50007`	Configures a BGP confederation identifier.
Step 4	**bgp confederation peers** *autonomous-system* [*autonomous-system* ...] **Example:** `Device(config)# bgp confederation peers 51000 51001 51002`	Specifies the autonomous systems that belong to the confederation and that will be treated as special EBGP peers.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ip bgp neighbor** **Example:** `Device# show ip bgp neighbor`	Verifies the configuration.
Step 7	**show ip bgp network** **Example:** `Device# show ip bgp network`	Verifies the configuration.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Route Reflectors

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters the global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 2	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 101`	Enters BGP router configuration mode.
Step 3	**neighbor** {*ip-address* | *peer-group-name*} **route-reflector-client** **Example:** `Device(config-router)# neighbor 172.16.70.24 route-reflector-client`	Configures the local router as a BGP route reflector and the specified neighbor as a client.
Step 4	**bgp cluster-id** *cluster-id* **Example:** `Device(config-router)# bgp cluster-id 10.0.1.2`	(Optional) Configures the cluster ID if the cluster has more than one route reflector.
Step 5	**no bgp client-to-client reflection** **Example:** `Device(config-router)# no bgp client-to-client reflection`	(Optional) Disables client-to-client route reflection. By default, the routes from a route reflector client are reflected to other clients. However, if the clients are fully meshed, the route reflector does not need to reflect routes to clients.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show ip bgp** **Example:** `Device# show ip bgp`	Verifies the configuration. Displays the originator ID and the cluster-list attributes.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Route Dampening

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 100`	Enters BGP router configuration mode.
Step 3	**bgp dampening** **Example:** `Device(config-router)# bgp dampening`	Enables BGP route dampening.
Step 4	**bgp dampening** *half-life reuse suppress max-suppress* [**route-map** *map*] **Example:** `Device(config-router)# bgp dampening 30 1500 10000 120`	(Optional) Changes the default values of route dampening factors.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask* [**longer-prefix**]}] **Example:** `Device# show ip bgp flap-statistics`	(Optional) Monitors the flaps of all paths that are flapping. The statistics are deleted when the route is not suppressed and is stable.
Step 7	**show ip bgp dampened-paths** **Example:** `Device# show pi bgp dampened-paths`	(Optional) Displays the dampened routes, including the time remaining before they are suppressed.
Step 8	**clear ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask* [**longer-prefix**]} **Example:**	(Optional) Clears BGP flap statistics to make it less likely that a route will be dampened.

	Command or Action	Purpose
	`Device# clear ip bgp flap-statistics`	
Step 9	**clear ip bgp dampening** **Example:** `Device# clear ip bgp dampening`	(Optional) Clears route dampening information, and unsuppress the suppressed routes.
Step 10	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring and Maintaining BGP

You can remove all contents of a particular cache, table, or database. This might be necessary when the contents of the particular structure have become or are suspected to be invalid.

You can display specific statistics, such as the contents of BGP routing tables, caches, and databases. You can use the information to get resource utilization and solve network problems. You can also display information about node reachability and discover the routing path your device's packets are taking through the network.

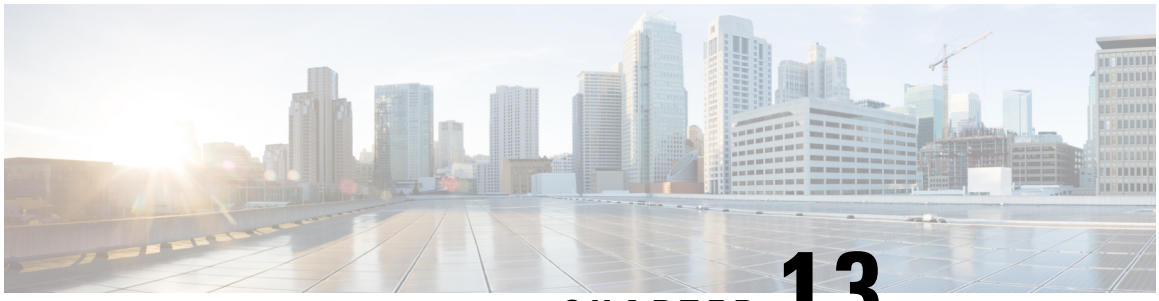
The table given below lists the privileged EXEC commands for clearing and displaying BGP.

Table 26: IP BGP Clear and Show Commands

clear ip bgp *address*	Resets a particular BGP connection.
clear ip bgp *	Resets all BGP connections.
clear ip bgp peer-group *tag*	Removes all members of a BGP peer group.
show ip bgp *prefix*	Displays peer groups and peers not in peer groups to wh has been advertised. Also displays prefix attributes such hop and the local prefix.
show ip bgp cidr-only	Displays all BGP routes that contain subnet and superne masks.
show ip bgp community [*community-number*] [**exact**]	Displays routes that belong to the specified communitie
show ip bgp community-list *community-list-number* [**exact-match**]	Displays routes that are permitted by the community lis
show ip bgp filter-list *access-list-number*	Displays routes that are matched by the specified AS pa
show ip bgp inconsistent-as	Displays the routes with inconsistent originating autonon
show ip bgp regexp *regular-expression*	Displays the routes that have an AS path that matches t regular expression entered on the command line.

show ip bgp	Displays the contents of the BGP routing table.
show ip bgp neighbors [*address*]	Displays detailed information on the BGP and TCP connec individual neighbors.
show ip bgp neighbors [*address*] [**advertised-routes** | **dampened-routes** | **flap-statistics** | **paths** *regular-expression* | **received-routes** | **routes**]	Displays routes learned from a particular BGP neighbor.
show ip bgp paths	Displays all BGP paths in the database.
show ip bgp peer-group [*tag*] [**summary**]	Displays information about BGP peer groups.
show ip bgp summary	Displays the status of all BGP connections.

The **bgp log-neighbor changes** command is enabled by default. It allows to log messages that are generated when a BGP neighbor resets, comes up, or goes down.



CHAPTER 13

Implementing Multiprotocol BGP for IPv6

This module describes how to configure multiprotocol Border Gateway Protocol (BGP) for IPv6. BGP is an Exterior Gateway Protocol (EGP) used mainly to connect separate routing domains that contain independent routing policies (autonomous systems). Connecting to a service provider for access to the Internet is a common use for BGP. BGP can also be used within an autonomous system and this variation is referred to as internal BGP (iBGP). Multiprotocol BGP is an enhanced BGP that carries routing information for multiple network layer protocol address families, for example, IPv6 address family and for IP multicast routes. All BGP commands and routing policy capabilities can be used with multiprotocol BGP.

Information About Implementing Multiprotocol BGP for IPv6

Multiprotocol BGP Extensions for IPv6

Multiprotocol BGP is the supported Exterior Gateway Protocol (EGP) for IPv6. Multiprotocol BGP extensions for IPv6 supports many of the same features and functionality as IPv4 BGP. IPv6 enhancements to multiprotocol BGP include support for an IPv6 address family and Network Layer Reachability Information (NLRI) and next hop (the next device in the path to the destination) attributes that use IPv6 addresses.

IPv6 Multiprotocol BGP Peering Using a Link-Local Address

The IPv6 multiprotocol BGP can be configured between two IPv6 devices (peers) using link-local addresses. For this function to work, you must identify the interface for the neighbor by using the **neighbor update-source** command, and you must configure a route map to set an IPv6 global next hop.

Multiprotocol BGP for the IPv6 Multicast Address Family

The multiprotocol BGP for the IPv6 multicast address family feature provides multicast BGP extensions for IPv6 and supports the same features and functionality as IPv4 BGP. IPv6 enhancements to multicast BGP

include support for an IPv6 multicast address family and network layer reachability information (NLRI) and next hop (the next router in the path to the destination) attributes that use IPv6 addresses.

Multicast BGP is an enhanced BGP that allows the deployment of interdomain IPv6 multicast. Multiprotocol BGP carries routing information for multiple network layer protocol address families; for example, IPv6 address family and for IPv6 multicast routes. The IPv6 multicast address family contains routes used for RPF lookup by the IPv6 PIM protocol, and multicast BGP IPv6 provides for interdomain transport of the same. Users must use multiprotocol BGP for IPv6 multicast when using IPv6 multicast with BGP because the unicast BGP learned routes will not be used for IPv6 multicast.

Multicast BGP functionality is provided through a separate address family context. A subsequent address family identifier (SAFI) provides information about the type of the network layer reachability information that is carried in the attribute. Multiprotocol BGP unicast uses SAFI 1 messages, and multiprotocol BGP multicast uses SAFI 2 messages. SAFI 1 messages indicate that the routes are usable only for IP unicast, not IP multicast. Because of this functionality, BGP routes in the IPv6 unicast RIB must be ignored in the IPv6 multicast RPF lookup.

A separate BGP routing table is maintained to configure incongruent policies and topologies (for example, IPv6 unicast and multicast) by using IPv6 multicast RPF lookup. Multicast RPF lookup is very similar to the IP unicast route lookup.

No MRIB is associated with the IPv6 multicast BGP table. However, IPv6 multicast BGP operates on the unicast IPv6 RIB when needed. Multicast BGP does not insert or update routes into the IPv6 unicast RIB.

Nonstop Forwarding and Graceful Restart for MP-BGP IPv6 Address Family

The graceful restart capability is supported for IPv6 BGP unicast, multicast, and VPNv6 address families, enabling Cisco nonstop forwarding (NSF) functionality for BGP IPv6. The BGP graceful restart capability allows the BGP routing table to be recovered from peers without keeping the TCP state.

NSF continues forwarding packets while routing protocols converge, therefore avoiding a route flap on switchover. Forwarding is maintained by synchronizing the FIB between the active and standby RP. On switchover, forwarding is maintained using the FIB. The RIB is not kept synchronized; therefore, the RIB is empty on switchover. The RIB is repopulated by the routing protocols and subsequently informs FIB about RIB convergence by using the NSF_RIB_CONVERGED registry call. The FIB tables are updated from the RIB, removing any stale entries. The RIB starts a failsafe timer during RP switchover, in case the routing protocols fail to notify the RIB of convergence.

The Cisco BGP address family identifier (AFI) model is designed to be modular and scalable, and to support multiple AFI and subsequent address family identifier (SAFI) configurations.

How to Implement Multiprotocol BGP for IPv6

Configuring an IPv6 BGP Routing Process and BGP Router ID

Perform this task to configure an IPv6 BGP routing process and an optional BGP router ID for a BGP-speaking device.

BGP uses a router ID to identify BGP-speaking peers. The BGP router ID is 32-bit value that is often represented by an IPv4 address. By default, the router ID is set to the IPv4 address of a loopback interface on the device. If no loopback interface is configured on the device, then the software chooses the highest IPv4 address configured to a physical interface on the device to represent the BGP router ID.

When configuring BGP on a device that is enabled only for IPv6 (that is, the device does not have an IPv4 address), you must manually configure the BGP router ID for the device. The BGP router ID, which is represented as a 32-bit value using an IPv4 address syntax, must be unique to the BGP peers of the device.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **no bgp default ipv4-unicast**
5. **bgp router-id** *ip-address*
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Configures a BGP routing process, and enters router configuration mode for the specified routing process.
Step 4	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process specified in the previous step. **Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured with the **neighbor remote-as** command unless you configure the **no bgp default ipv4-unicast** command before configuring the **neighbor remote-as** command.
Step 5	**bgp router-id** *ip-address* **Example:** `Device(config-router)# bgp router-id 192.168.99.70`	(Optional) Configures a fixed 32-bit router ID as the identifier of the local device running BGP. **Note** Configuring a router ID using the **bgp router-id** command resets all active BGP peering sessions.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

Configuring IPv6 Multiprotocol BGP Between Two Peers

By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types, as shown for IPv6 prefixes.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **neighbor** {*ip-address* | *ipv6-address* [**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number* ...]
5. **address-family ipv6** [**unicast** | **multicast**]
6. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* **%**} **activate**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *ipv6-address* [**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number* ...] **Example:**	Adds the IPv6 address of the neighbor in the specified autonomous system to the IPv6 multiprotocol BGP neighbor table of the local device.

	Command or Action	Purpose
	`Device(config-router)# neighbor 2001:DB8:0:CC00::1 remote-as 64600`	
Step 5	**address-family ipv6** [**unicast** | **multicast**] **Example:** `Device(config-router)# address-family ipv6`	Specifies the IPv6 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv6 unicast address family. By default, the device is placed in configuration mode for the IPv6 unicast address family if a keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 6	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* **%**} **activate** **Example:** `Device(config-router-af)# neighbor 2001:DB8:0:CC00::1 activate`	Enables the neighbor to exchange prefixes for the IPv6 address family with the local device.
Step 7	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

Configuring IPv6 Multiprotocol BGP Between Two Peers Using Link-Local Addresses

By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types, as shown for IPv6 prefixes.

By default, route maps that are applied in router configuration mode using the **neighbor route-map** command are applied to only IPv4 unicast address prefixes. Route maps for other address families must be applied in address family configuration mode using the **neighbor route-map** command, as shown for the IPv6 address family. The route maps are applied either as the inbound or outbound routing policy for neighbors under the specified address family. Configuring separate route maps under each address family type simplifies managing complicated or different policies for each address family.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **remote-as** *as-number*

5. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **update-source** *interface-type interface-number*
6. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**
7. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*} activate
8. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*} **route-map** *map-name* {**in** | **out**
9. **exit**
10. **exit**
11. **route-map** *map-tag* [**permit** | **deny**] [*sequence-number*]
12. **match ipv6 address** {**prefix-list** *prefix-list-name* | *access-list-name*
13. **set ipv6 next-hop** *ipv6-address* [*link-local-address*] [**peer-address**
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 remote-as 64600`	Adds the link-local IPv6 address of the neighbor in the specified remote autonomous system to the IPv6 multiprotocol BGP neighbor table of the local router. • The *ipv6-address*argument in the **neighbor remote-as**command must be a link-local IPv6 address in the form documented in RFC 2373 where the address is specified in hexadecimal using 16-bit values between colons.
Step 5	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 update-source gigabitethernet 0/0/0`	Specifies the link-local address over which the peering is to occur. • If there are multiple connections to the neighbor and you do not specify the neighbor interface by using the *interface-type* and *interface-number*arguments in the **neighbor update-source** command, a TCP connection cannot be established with the neighbor using link-local addresses.

	Command or Action	Purpose
Step 6	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6** **Example:** `Device(config-router)# address-family ipv6`	Specifies the IPv6 address family, and enters address family configuration mode. • The **unicast** keyword specifies the IPv6 unicast address family. By default, the router is placed in configuration mode for the IPv6 unicast address family if the **unicast** keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 7	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*} activate **Example:** `Device(config-router-af)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 activate`	Enables the neighbor to exchange prefixes for the IPv6 address family with the local router using the specified link-local addresses.
Step 8	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*} **route-map** *map-name* {**in** | **out** **Example:** `Device(config-router-af)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 route-map nh6 out`	Applies a route map to incoming or outgoing routes.
Step 9	**exit** **Example:** `Device(config-router-af)# exit`	Exits address family configuration mode, and returns to router configuration mode.
Step 10	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode, and returns to global configuration mode.
Step 11	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map nh6 permit 10`	Defines a route map and enters route-map configuration mode.
Step 12	**match ipv6 address** {**prefix-list** *prefix-list-name* | *access-list-name* **Example:** `Device(config-route-map)# match ipv6 address prefix-list list1`	Distributes any routes that have a destination IPv6 network number address permitted by a prefix list, or performs policy routing on packets.

	Command or Action	Purpose
Step 13	**set ipv6 next-hop** *ipv6-address* [*link-local-address*] [**peer-address** **Example:** `Device(config-route-map)# set ipv6 next-hop 2001:DB8::1`	Overrides the next hop advertised to the peer for IPv6 packets that pass a match clause of a route map for policy routing. • The *ipv6-address* argument specifies the IPv6 global address of the next hop. It need not be an adjacent router. • The *link-local-address*argument specifies the IPv6 link-local address of the next hop. It must be an adjacent router. **Note** The route map sets the IPv6 next-hop addresses (global and link-local) in BGP updates. If the route map is not configured, the next-hop address in the BGP updates defaults to the unspecified IPv6 address (::), which is rejected by the peer. If you specify only the global IPv6 next-hop address (the *ipv6-address* argument) with the **set ipv6 next-hop**command after specifying the neighbor interface (the *interface-type* argument) with the **neighbor update-source** command in Step 5, the link-local address of the interface specified with the *interface-type* argument is included as the next-hop in the BGP updates. Therefore, only one route map that sets the global IPv6 next-hop address in BGP updates is required for multiple BGP peers that use link-local addresses.
Step 14	**end** **Example:** `Device(config-route-map)# end`	Exits route-map configuration mode and returns to privileged EXEC mode.

Troubleshooting Tips

If peering is not established by this task, it may be because of a missing route map **set ipv6 next-hop** command. Use the **debug bgp ipv6 update** command to display debugging information on the updates to help determine the state of the peering.

Configuring an IPv6 Multiprotocol BGP Peer Group

- By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types, as shown for IPv6 prefixes.

- By default, peer groups that are defined in router configuration mode using the **neighbor peer-group**command exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, you must activate peer groups using the **neighbor activate** command in address family configuration mode for the other prefix types, as shown for IPv6 prefixes.
- Members of a peer group automatically inherit the address prefix configuration of the peer group.
- IPv4 active neighbors cannot exist in the same peer group as active IPv6 neighbors. Create separate peer groups for IPv4 peers and IPv6 peers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **neighbor** *peer-group-name* **peer-group**
5. **neighbor** {*ip-address* | *ipv6-address*[**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number ...*]
6. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**
7. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* **%**} **activate**
8. **neighbor** *ip-address* | *ipv6-address*} **send-label**
9. **neighbor** {*ip-address* | *ipv6-address*} **peer-group** *peer-group-name*
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified BGP routing process.
Step 4	**neighbor** *peer-group-name* **peer-group** **Example:** `Device(config-router)# neighbor group1 peer-group`	Creates a multiprotocol BGP peer group.

	Command or Action	Purpose
Step 5	**neighbor** {*ip-address* | *ipv6-address*[**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number* ...] **Example:** `Device(config-router)# neighbor 2001:DB8:0:CC00::1 remote-as 64600`	Adds the IPv6 address of the neighbor in the specified autonomous system to the IPv6 multiprotocol BGP neighbor table of the local router.
Step 6	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6** **Example:** `Device(config-router)# address-family ipv6 unicast`	Specifies the IPv6 address family, and enters address family configuration mode. • The **unicast** keyword specifies the IPv6 unicast address family. By default, the device is placed in configuration mode for the IPv6 unicast address family if the **unicast** keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 7	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* **%**} **activate** **Example:** `Device(config-router-af)# neighbor 2001:DB8:0:CC00::1 activate`	Enables the neighbor to exchange prefixes for the specified family type with the neighbor and the local router. • To avoid extra configuration steps for each neighbor, use the **neighbor activate** command with the *peer-group-name* argument as an alternative in this step.
Step 8	**neighbor** *ip-address* | *ipv6-address*} **send-label** **Example:** `Device(config-router-af)# neighbor 192.168.99.70 send-label`	Advertises the capability of the device to send MPLS labels with BGP routes. • In IPv6 address family configuration mode, this command enables binding and advertisement of aggregate labels when advertising IPv6 prefixes in BGP.
Step 9	**neighbor** {*ip-address* | *ipv6-address*} **peer-group** *peer-group-name* **Example:** `Device(config-router-af)# neighbor 2001:DB8:0:CC00::1 peer-group group1`	Assigns the IPv6 address of a BGP neighbor to a peer group.
Step 10	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode, and returns to privileged EXEC mode.

Configuring a Route Map for IPv6 Multiprotocol BGP Prefixes

- By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types, as shown for IPv6 prefixes.
- By default, route maps that are applied in router configuration mode using the **neighbor route-map** command are applied to only IPv4 unicast address prefixes. Route maps for other address families must be applied in address family configuration mode using the **neighbor route-map** command, as shown for the IPv6 address family. The route maps are applied either as the inbound or outbound routing policy for neighbors under the specified address family. Configuring separate route maps under each address family type simplifies managing complicated or different policies for each address family.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **neighbor** {*ip-address* | *ipv6-address*[**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number ...*]
5. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**]
6. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* **%**} **activate**
7. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* [**%**]} **route-map** *map-name* {**in** | **out**}
8. **exit**
9. **exit**
10. **route-map** *map-tag* [**permit** | **deny**] [*sequence-number*]
11. **match ipv6 address** {**prefix-list** *prefix-list-name* | *access-list-name*}
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.

	Command or Action	Purpose
Step 4	**neighbor** {*ip-address* | *ipv6-address*[**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number* ...] **Example:** `Device(config-router)# neighbor 2001:DB8:0:cc00::1 remote-as 64600`	Adds the link-local IPv6 address of the neighbor in the specified remote autonomous system to the IPv6 multiprotocol BGP neighbor table of the local device.
Step 5	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**] **Example:** `Device(config-router)# address-family ipv6`	Specifies the IPv6 address family, and enters address family configuration mode. • The **unicast** keyword specifies the IPv6 unicast address family. By default, the device is placed in configuration mode for the IPv6 unicast address family if the **unicast** keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 6	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* **%**} **activate** **Example:** `Device(config-router-af)# neighbor 2001:DB8:0:cc00::1 activate`	Enables the neighbor to exchange prefixes for the IPv6 address family with the local device using the specified link-local addresses.
Step 7	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* [**%**]} **route-map** *map-name* {**in** | **out**} **Example:** `Device(config-router-af)# neighbor 2001:DB8:0:cc00::1 route-map rtp in`	Applies a route map to incoming or outgoing routes. • Changes to the route map will not take effect for existing peers until the peering is reset or a soft reset is performed. Using the **clear bgp ipv6** command with the **soft** and **in** keywords will perform a soft reset.
Step 8	**exit** **Example:** `Device(config-router-af)# exit`	Exits address family configuration mode, and returns to router configuration mode.
Step 9	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode, and returns to global configuration mode.
Step 10	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map rtp permit 10`	Defines a route map and enters route-map configuration mode. • Follow this step with a **match** command.

	Command or Action	Purpose
Step 11	**match ipv6 address** {**prefix-list** *prefix-list-name* | *access-list-name*} **Example:** `Device(config-route-map)# match ipv6 address prefix-list list1`	Distributes any routes that have a destination IPv6 network number address permitted by a prefix list, or performs policy routing on packets.
Step 12	**end** **Example:** `Device(config-route-map)# end`	Exits route-map configuration mode and returns to privileged EXEC mode.

Redistributing Prefixes into IPv6 Multiprotocol BGP

Redistribution is the process of redistributing, or injecting, prefixes from one routing protocol into another routing protocol. This task explains how to inject prefixes from a routing protocol into IPv6 multiprotocol BGP. Specifically, prefixes that are redistributed into IPv6 multiprotocol BGP using the **redistribute** router configuration command are injected into the IPv6 unicast database.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**]
5. **redistribute bgp** [*process-id*] [**metric** *metric-value*] [**route-map** *map-name*]
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified BGP routing process.

	Command or Action	Purpose
Step 4	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**] **Example:** `Device(config-router)# address-family ipv6`	Specifies the IPv6 address family, and enters address family configuration mode. • The **unicast** keyword specifies the IPv6 unicast address family. By default, the device is placed in configuration mode for the IPv6 unicast address family if a keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 5	**redistribute bgp** [*process-id*] [**metric** *metric-value*] [**route-map** *map-name*] **Example:** `Device(config-router-af)# redistribute bgp 64500 metric 5`	Redistributes IPv6 routes from one routing domain into another routing domain.
Step 6	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode, and returns to privileged EXEC mode.

Advertising Routes into IPv6 Multiprotocol BGP

By default, networks that are defined in router configuration mode using the **network** command are injected into the IPv4 unicast database. To inject a network into another database, such as the IPv6 BGP database, you must define the network using the **network** command in address family configuration mode for the other database, as shown for the IPv6 BGP database.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**]
5. **network** {*network-number* [**mask** *network-mask*] | *nsap-prefix*} [**route-map** *map-tag*]
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified BGP routing process.
Step 4	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**] **Example:** `Device(config-router)# address-family ipv6 unicast`	Specifies the IPv6 address family, and enters address family configuration mode. • The **unicast** keyword specifies the IPv6 unicast address family. By default, the device is placed in configuration mode for the IPv6 unicast address family if a keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 5	**network** {*network-number* [**mask** *network-mask*] | *nsap-prefix*} [**route-map** *map-tag*] **Example:** `Device(config-router-af)# network 2001:DB8::/24`	Advertises (injects) the specified prefix into the IPv6 BGP database (the routes must first be found in the IPv6 unicast routing table). • The prefix is injected into the database for the address family specified in the previous step. • Routes are tagged from the specified prefix as "local origin." • The *ipv6-prefix* argument in the **network** command must be in the form documented in RFC 2373 where the address is specified in hexadecimal using 16-bit values between colons. • The *prefix-length* argument is a decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash mark must precede the decimal value.
Step 6	**exit** **Example:** `Device(config-router-af)# exit`	Exits address family configuration mode, and returns the device to router configuration mode. • Repeat this step to exit router configuration mode and return the device to global configuration mode.

Advertising IPv4 Routes Between IPv6 BGP Peers

If an IPv6 network is connecting two separate IPv4 networks, IPv6 can be used to advertise the IPv4 routes. Configure the peering using the IPv6 addresses within the IPv4 address family. Set the next hop with a static route or with an inbound route map because the advertised next hop will usually be unreachable. Advertising IPv6 routes between two IPv4 peers is also possible using the same model.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **neighbor** *peer-group-name* **peer-group**
5. **neighbor** {*ip-address* | *ipv6-address*[**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number ...*]
6. **address-family ipv4** [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*]
7. **neighbor** *ipv6-address* **peer-group** *peer-group-name*
8. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* [**%**]} **route-map** *map-name* {**in** | **out**}
9. **exit**
10. **exit**
11. **route-map** *map-tag* [**permit** | **deny**] [*sequence-number*]
12. **set ip next-hop** *ip-address* [*... ip-address*] [**peer-address**]
13. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** *peer-group-name* **peer-group** **Example:** `Device(config-router)# neighbor 6peers peer-group`	Creates a multiprotocol BGP peer group.

	Command or Action	Purpose
Step 5	**neighbor** {*ip-address* | *ipv6-address*[**%**] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number* ...] **Example:** `Device(config-router)# neighbor 6peers remote-as 65002`	Adds the IPv6 address of the neighbor in the specified autonomous system to the IPv6 multiprotocol BGP neighbor table of the local device.
Step 6	**address-family ipv4** [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] **Example:** `Device(config-router)# address-family ipv4`	Enters address family configuration mode to configure a routing session using standard IPv4 address prefixes.
Step 7	**neighbor** *ipv6-address* **peer-group** *peer-group-name* **Example:** `Device(config-router-af)# neighbor 2001:DB8:1234::2 peer-group 6peers`	Assigns the IPv6 address of a BGP neighbor to a peer group.
Step 8	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address* [**%**]} **route-map** *map-name* {**in** | **out**} **Example:** `Device(config-router-af)# neighbor 6peers route-map rmap out`	Applies a route map to incoming or outgoing routes. • Changes to the route map will not take effect for existing peers until the peering is reset or a soft reset is performed. Using the **clear bgp ipv6** command with the **soft** and **in** keywords will perform a soft reset.
Step 9	**exit** **Example:** `Device(config-router-af)# exit`	Exits address family configuration mode, and returns the device to router configuration mode.
Step 10	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode, and returns the device to global configuration mode.
Step 11	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map rmap permit 10`	Defines a route map and enters route-map configuration mode.
Step 12	**set ip next-hop** *ip-address* [... *ip-address*] [**peer-address**] **Example:** `Device(config-route-map)# set ip next-hop 10.21.8.10`	Overrides the next hop advertised to the peer for IPv4 packets.

	Command or Action	Purpose
Step 13	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

Assigning BGP Administrative Distance for Multicast BGP Routes

Perform this task to specify an administrative distance for multicast BGP routes to be used in RPF lookups for comparison with unicast routes.

Caution Changing the administrative distance of BGP internal routes is not recommended. One problem that can occur is the accumulation of routing table inconsistencies, which can break routing.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**]
5. **distance bgp** *external-distance internal-distance local-distance*
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**] **Example:**	Specifies the IPv6 address family, and enters address family configuration mode.

	Command or Action	Purpose
	`Device(config-router)# address-family ipv6`	• The **unicast** keyword specifies the IPv6 unicast address family. By default, the router is placed in configuration mode for the IPv6 unicast address family if the **unicast** keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 5	**distance bgp** *external-distance internal-distance local-distance* **Example:** `Device(config-router-af)# distance bgp 10 50 100`	Configures the administrative distance for BGP routes.
Step 6	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

Generating IPv6 Multicast BGP Updates

Perform this task to generate IPv6 multicast BGP updates that correspond to unicast IPv6 updates received from a peer.

The MBGP translate-update feature generally is used in an MBGP-capable router that peers with a customer site that has only a BGP-capable router; the customer site has not or cannot upgrade its router to an MBGP-capable image. Because the customer site cannot originate MBGP advertisements, the router with which it peers will translate the BGP prefixes into MBGP prefixes, which are used for multicast-source Reverse Path Forwarding (RPF) lookup.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**
5. **neighbor** *ipv6-address* **translate-update ipv6 multicast** [**unicast**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6** **Example:** `Device(config-router)# address-family ipv6`	Specifies the IPv6 address family, and enters address family configuration mode. • The **unicast** keyword specifies the IPv6 unicast address family. By default, the router is placed in configuration mode for the IPv6 unicast address family if the **unicast** keyword is not specified with the **address-family ipv6** command. • The **multicast** keyword specifies IPv6 multicast address prefixes.
Step 5	**neighbor** *ipv6-address* **translate-update ipv6 multicast** [**unicast** **Example:** `Device(config-router-af)# neighbor 2001:DB8::2 translate-update ipv6 multicast`	Generates multiprotocol IPv6 BGP updates that correspond to unicast IPv6 updates received from a peer.
Step 6	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

Configuring the IPv6 BGP Graceful Restart Capability

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **bgp graceful-restart** [**restart-time** *seconds* | **stalepath-time** *seconds*] [**all**]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**bgp graceful-restart** [**restart-time** *seconds* | **stalepath-time** *seconds*] [**all**] **Example:** `Device(config-router)# bgp graceful-restart`	Enables the BGP graceful restart capability.
Step 5	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

Resetting IPv6 BGP Sessions

SUMMARY STEPS

1. **enable**
2. **clear bgp ipv6** {**unicast** | **multicast**} {***** | *autonomous-system-number* | *ip-address* | *ipv6-address* | **peer-group** *peer-group-name*} [**soft**] [**in** | **out**]
3. **clear bgp ipv6** {**unicast** | **multicast**} **external** [**soft**] [**in** | **out**]
4. **clear bgp ipv6** {**unicast** | **multicast**} **peer-group** *name*
5. **clear bgp ipv6** {**unicast** | **multicast**} **dampening** [*ipv6-prefix*/*prefix-length*]
6. **clear bgp ipv6** {**unicast** | **multicast**} **flap-statistics** [*ipv6-prefix*/*prefix-length* | **regexp** *regexp* | **filter-list** *list*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**clear bgp ipv6** {**unicast** | **multicast**} {***** | *autonomous-system-number* | *ip-address* | *ipv6-address* | **peer-group** *peer-group-name*} [**soft**] [**in** | **out**] **Example:** `Device# clear bgp ipv6 unicast peer-group marketing soft out`	Resets IPv6 BGP sessions.
Step 3	**clear bgp ipv6** {**unicast** | **multicast**} **external** [**soft**] [**in** | **out**] **Example:** `Device# clear bgp ipv6 unicast external soft in`	Clears external IPv6 BGP peers.
Step 4	**clear bgp ipv6** {**unicast** | **multicast**} **peer-group** *name* **Example:** `Device# clear bgp ipv6 unicast peer-group marketing`	Clears all members of an IPv6 BGP peer group.
Step 5	**clear bgp ipv6** {**unicast** | **multicast**} **dampening** [*ipv6-prefix*/*prefix-length*] **Example:** `Device# clear bgp ipv6 unicast dampening 2001:DB8::/64`	Clears IPv6 BGP route dampening information and unsuppresses the suppressed routes.
Step 6	**clear bgp ipv6** {**unicast** | **multicast**} **flap-statistics** [*ipv6-prefix*/*prefix-length* | **regexp** *regexp* | **filter-list** *list*] **Example:** `Device# clear bgp ipv6 unicast flap-statistics filter-list 3`	Clears IPv6 BGP flap statistics.

Verifying the IPv6 Multiprotocol BGP Configuration

SUMMARY STEPS

1. **enable**
2. **show bgp ipv6 unicast** | **multicast**} [*ipv6-prefix*/*prefix-length*] [**longer-prefixes**] [**labels**]
3. **show bgp ipv6** {**unicast** | **multicast**} **summary**
4. **show bgp ipv6** {**unicast** | **multicast**} **dampening dampened-paths**
5. **debug bgp ipv6** {**unicast** | **multicast**} **dampening**[**prefix-list** *prefix-list-name*]

6. **debug bgp ipv6 unicast** | **multicast**} **updates**[*ipv6-address*] [**prefix-list** *prefix-list-name*] [**in**| **out**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show bgp ipv6 unicast** | **multicast**} [*ipv6-prefix*/*prefix-length*] [**longer-prefixes**] [**labels**] **Example:** `Device> show bgp ipv6 unicast`	(Optional) Displays entries in the IPv6 BGP routing table.
Step 3	**show bgp ipv6** {**unicast** | **multicast**} **summary** **Example:** `Device> show bgp ipv6 unicast summary`	(Optional) Displays the status of all IPv6 BGP connections.
Step 4	**show bgp ipv6** {**unicast** | **multicast**} **dampening dampened-paths** **Example:** `Device> show bgp ipv6 unicast dampening dampened-paths`	(Optional) Displays IPv6 BGP dampened routes.
Step 5	**debug bgp ipv6** {**unicast** | **multicast**} **dampening**[**prefix-list** *prefix-list-name*] **Example:** `Device# debug bgp ipv6 unicast dampening`	(Optional) Displays debugging messages for IPv6 BGP dampening packets. • If no prefix list is specified, debugging messages for all IPv6 BGP dampening packets are displayed.
Step 6	**debug bgp ipv6 unicast** | **multicast**} **updates**[*ipv6-address*] [**prefix-list** *prefix-list-name*] [**in**| **out**] **Example:** `Device# debug bgp ipv6 unicast updates`	(Optional) Displays debugging messages for IPv6 BGP update packets. • If an *ipv6-address* argument is specified, debugging messages for IPv6 BGP updates to the specified neighbor are displayed. • Use the **in** keyword to display debugging messages for inbound updates only. • Use the **out** keyword to display debugging messages for outbound updates only.

Configuration Examples for Implementing Multiprotocol BGP for IPv6

Example: Configuring a BGP Process, BGP Router ID, and IPv6 Multiprotocol BGP Peer

The following example enables IPv6 globally, configures a BGP process, and establishes a BGP router ID. Also, the IPv6 multiprotocol BGP peer 2001:DB8:0:CC00::1 is configured and activated.

```
Device> enable
Device# configure terminal
Device(config)# ipv6 unicast-routing
Device(config)# router bgp 65000
Device(config-router)# no bgp default ipv4-unicast
Device(config-router)# bgp router-id 192.168.99.70
Device(config-router)# neighbor 2001:DB8:0:CC00::1 remote-as 64600
Device(config-router)# address-family ipv6 unicast
Device(config-router-af)# neighbor 2001:DB8:0:CC00::1 activate
Device(config-router-af)# end
```

Example: Configuring an IPv6 Multiprotocol BGP Peer Using a Link-Local Address

The following example configures the IPv6 multiprotocol BGP peer FE80::XXXX:BFF:FE0E:A471 over Gigabit Ethernet interface 0/0/0 and sets the route map named nh6 to include the IPv6 next-hop global address of Gigabit Ethernet interface 0/0/0 in BGP updates. The IPv6 next-hop link-local address can be set by the nh6 route map (not shown in the following example) or from the interface specified by the **neighbor update-source** command (as shown in the following example).

```
Device> enable
Device# configure terminal
Device(config)# router bgp 65000
Device(config-router)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 remote-as 64600
Device(config-router)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 update-source
gigabitethernet 0/0/0
Device(config-router)# address-family ipv6
Device(config-router-af)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 activate
Device(config-router-af)# neighbor 2001:DB8:0000:0000:0000:0000:0000:0111 route-map nh6 out
Device(config-router-af)# exit
Device(config-router)# exit
Device(config)# route-map nh6 permit 10
Device(config-route-map)# match ipv6 address prefix-list list1
Device(config-route-map)# set ipv6 next-hop 2001:DB8:5y6::1
Device(config-route-map)# exit
Device(config)# ipv6 prefix-list list1 permit 2001:DB8:2Fy2::/48 le 128
Device(config)# ipv6 prefix-list list1 deny ::/0
Device(config)# end
```

Note If you specify only the global IPv6 next-hop address (the *ipv6-address* argument) with the **set ipv6 next-hop** command after specifying the neighbor interface (the *interface-type* argument) with the **neighbor update-source** command, the link-local address of the interface specified with the *interface-type* argument is included as the next hop in the BGP updates. Therefore, only one route map that sets the global IPv6 next-hop address in BGP updates is required for multiple BGP peers that use link-local addresses.

Example Configuring an IPv6 Multiprotocol BGP Peer Group

The following example configures the IPv6 multiprotocol BGP peer group named group1:

```
Device> enable
Device# configure terminal
Device(config)# router bgp 65000
Device(config-router)# no bgp default ipv4-unicast
Device(config-router)# neighbor group1 peer-group
Device(config-router)# neighbor group1 remote-as 100
Device(config-router)# neighbor group1 update-source Loopback0
Device(config-router)# neighbor 2001:DB8::1 peer-group group1
Device(config-router)# neighbor 2001:DB8:2:2 peer-group group1
Device(config-router)# address-family ipv6 multicast
Device(config-router-af)# neighbor 2001:DB8::1 activate
Device(config-router-af)# neighbor 2001:DB8:2:2 activate
Device(config-router-af)# exit-address-family
Device(config-router)# end
```

Example: Configuring a Route Map for IPv6 Multiprotocol BGP Prefixes

The following example configures the route map named rtp to permit IPv6 unicast routes from network 2001:DB8::/24 if they match the prefix list named list1:

```
Device> enable
Device# configure terminal
Device(config)# router bgp 64900
Device(config-router)# no bgp default ipv4-unicast
Device(config-router)# neighbor 2001:DB8:0:CC00::1 remote-as 64700
Device(config-router)# address-family ipv6 unicast
Device(config-router-af)# neighbor 2001:DB8:0:CC00::1 activate
Device(config-router-af)# neighbor 2001:DB8:0:CC00::1 route-map rtp in
Device(config-router-af)# exit
Device(config)# ipv6 prefix-list cisco seq 10 permit 2001:DB8::/24
Device(config)# route-map rtp permit 10
Device(config-route-map)# match ipv6 address prefix-list list1
Device(config-route-map)# end
```

Example Redistributing Prefixes into IPv6 Multiprotocol BGP

The following example redistributes BGP routes into the IPv6 multicast database of the local router:

```
router bgp 64900
 no bgp default ipv4-unicast
```

```
address-family ipv6 multicast
 redistribute BGP
```

Example: Advertising Routes into IPv6 Multiprotocol BGP

The following example injects the IPv6 network 2001:DB8::/24 into the IPv6 unicast database of the local device. (BGP checks that a route for the network exists in the IPv6 unicast database of the local device before advertising the network.)

```
Device> enable
Device# configure terminal
Device(config)# router bgp 65000
Device(config-router)# no bgp default ipv4-unicast
Device(config-router)# address-family ipv6 unicast
Device(config-router-af)# network 2001:DB8::/24
Device(config-router-af)# end
```

Example: Advertising IPv4 Routes Between IPv6 Peers

The following example advertises IPv4 routes between IPv6 peers when the IPv6 network is connecting two separate IPv4 networks. Peering is configured using IPv6 addresses in the IPv4 address family configuration mode. The inbound route map named rmap sets the next hop because the advertised next hop is likely to be unreachable.

```
Device> enable
Device# configure terminal
Device(config)# router bgp 65000
Device(config-router)# neighbor 6peers peer-group
Device(config-router)# neighbor 2001:DB8:1234::2 remote-as 65002
Device(config-router)# address-family ipv4
Device(config-router)# neighbor 6peers activate
Device(config-router)# neighbor 6peers soft-reconfiguration inbound
Device(config-router)# neighbor 2001:DB8:1234::2 peer-group 6peers
Device(config-router)# neighbor 2001:DB8:1234::2 route-map rmap in
Device(config-router)# exit
Device(config)# route-map rmap permit 10
Device(config-route-map)# set ip next-hop 10.21.8.10
Device(config-route-map)# end
```

Additional References for Implementing Multiprotocol BGP for IPv6

Standards and RFCs

RFCs	Title
RFC 2545	*Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 4007	*IPv6 Scoped Address Architecture*

RFCs	Title
RFC 4364	BGP MPLS/IP Virtual Private Networks (VPNs)
RFC 4382	MPLS/BGP Layer 3 Virtual Private Network (VPN) Management Information Base
RFC 4659	BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN
RFC 4724	*Graceful Restart Mechanism for BGP*

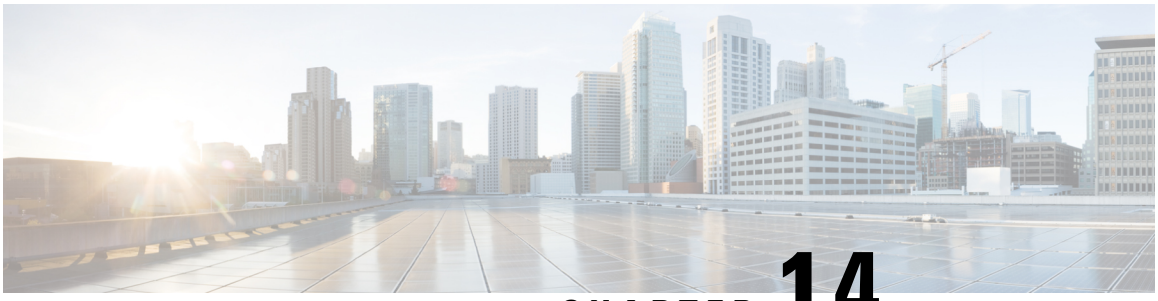
Feature Information for Implementing Multiprotocol BGP for IPv6

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use the Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 27: Feature Information for Implementing Multiprotocol BGP for IPv6

Feature Name	Releases	Feature Information
Multiprotocol BGP for IPv6	Cisco IOS XE Everest 16.6.1	Multiprotocol BGP extensions for IPv6 supports the same features and functionality as IPv4 BGP.



CHAPTER 14

Configuring IS-IS Routing

Information About IS-IS Routing

Integrated Intermediate System-to-Intermediate System (IS-IS) is an ISO dynamic routing protocol (described in ISO 105890). To enable IS-IS you should create an IS-IS routing process and assign it to a specific interface, rather than to a network. You can specify more than one IS-IS routing process per Layer 3 device by using the multiarea IS-IS configuration syntax. You should then configure the parameters for each instance of the IS-IS routing process.

Small IS-IS networks are built as a single area that includes all the devices in the network. As the network grows larger, the netwok reorganizes itself into a backbone area made up of all the connected set of Level 2 devices still connected to their local areas. Within a local area, devices know how to reach all system IDs. Between areas, devices know how to reach the backbone, and the backbone devices know how to reach other areas.

Devices establish Level 1 adjacencies to perform routing within a local area (station routing). Devices establish Level 2 adjacencies to perform routing between Level 1 areas (area routing).

A single Cisco device can participate in routing in up to 29 areas and can perform Level 2 routing in the backbone. In general, each routing process corresponds to an area. By default, the first instance of the routing process that is configured performs both Level 1 and Level 2 routing. You can configure additional device instances, which are automatically treated as Level 1 areas. You must configure the parameters for each instance of the IS-IS routing process individually.

For IS-IS multiarea routing, you can configure only one process to perform Level 2 routing, although you can define up to 29 Level 1 areas for each Cisco unit. If Level 2 routing is configured on any process, all additional processes are automatically configured as Level 1. You can configure this process to perform Level 1 routing at the same time. If Level 2 routing is not desired for a device instance, remove the Level 2 capability using the **is-type** command in global configuration mode. Use the **is-type** command also to configure a different device instance as a Level 2 device.

IS-IS Authentication

To prevent unauthorized devices from injecting false routing information into the link-state database, you can either set a plain text password for each interface and an area password for each IS-IS area, or you can configure an IS-IS authentication.

Plain text passwords do not provide security against unauthorized users. You can configure a plain text password to prevent unauthorized networking devices from forming adjacencies with the router. The password is exchanged as plain text and is visible to agents having access to view the IS-IS packets.

The new style of IS-IS authentication provides the following advantages over the plain text password configuration commands:

- Passwords are encrypted when the software configuration is displayed.
- Passwords are easier to manage and change.
- Passwords can be changed to new passwords without disrupting network operations.
- Authentication transitions which are nondisruptive.

Authentication modes (IS-IS authentication or plain text password) can either be configured on a given scope (IS-IS instance or interface) or level, but not both. However, different modes can be configured for different scopes or levels. In case mixed modes are configured, different keys must be used for different modes to ensure that the encrypted passwords in the protocol data units (PDUs) are not compromised.

Clear Text Authentication

IS-IS clear text authentication provides the same functionality provided by the **area-password** or **domain-password** command.

HMAC-MD5 Authentication

IS-IS supports message digest algorithm 5 (MD5) authentication, which is more secure than clear text authentication.

Hashed Message Authentication Code (HMAC) is a mechanism for message authentication codes (MACs) using cryptographic hash functions. HMAC-MD5 authentication adds an HMAC-MD5 digest to each IS-IS PDU. The digest allows authentication at the IS-IS routing protocol level, which prevents unauthorized routing messages from being injected into the network routing domain.

The following are the benefits of HMAC-MD5 authentication:

- Passwords can be changed to new passwords without disrupting routing messages.
- Authentication transitions which are nondisruptive. The device accepts PDUs with either no authentication information or stale authentication information and sends PDUs with current authentication information. These transitions are useful when migrating from no authentication to some type of authentication, when changing the authentication type, and when changing the authentication keys.

HMAC-SHA Authentication

IS-IS supports Secure Hash Algorithm (SHA) authentication, that is, SHA-1, SHA-256, SHA-384, and SHA-512, which is more secure than MD5 authentication or clear text authentication.

When you enable the HMAC-SHA authentication method, a shared secret key is configured on all the devices that are connected on a common network. For each packet, this key is used to generate and verify a message digest that gets added to the packet. The message digest is a one-way function of the packet and the secret key.

Hitless Upgrade

Before you migrate from using one type of security authentication to another, you must do the following:

1. All the devices must be loaded with the new image that supports the new authentication type. The devices will continue to use the original authentication method until all the devices have been loaded with the new image that supports the new authentication method, and all the devices have been configured to use the new authentication method.
2. Add a key chain with both the current key and a new key. For example when migrating from HMAC-MD5 to HMAC-SHA1-20, the current key is HMAC-MD5, and the new key is HMAC-SHA1-20. Ensure that the current key has a later end date for the send-lifetime field than the new key so that IS-IS continues to send the current key. Set the accept-lifetime value of both the keys to infinite so that IS-IS accepts both the keys.
3. After step 2 is completed, for all the devices in a link or area the current key can be removed from the key chain.

Nonstop Forwarding Awareness

The integrated IS-IS Nonstop Forwarding (NSF) Awareness feature is supported for IPv4G. The feature allows customer premises equipment (CPE) devices that are NSF-aware to help NSF-capable devices perform nonstop forwarding of packets. The local device is not necessarily performing NSF, but its NSF awareness capability allows the integrity and accuracy of the routing database and the link-state database on the neighboring NSF-capable device to be maintained during the switchover process.

The integrated IS-IS Nonstop Forwarding (NSF) Awareness feature is automatically enabled and requires no configuration.

IS-IS Global Parameters

The following are the optional IS-IS global parameters that you can configure:

- You can force a default route into an IS-IS routing domain by configuring a default route that is controlled by a route map. You can also specify the other filtering options that are configurable under a route map.
- You can configure the device to ignore IS-IS link-state packets (LSPs) that are received with internal checksum errors, or to purge corrupted LSPs, and cause the initiator of the LSP to regenerate it.
- You can assign passwords to areas and domains.
- You can create aggregate addresses that are represented in the routing table by a summary address (based on route summarization). Routes learned from other routing protocols can also be summarized. The metric used to advertise the summary is the smallest metric of all the specific routes.
- You can set an overload bit.
- You can configure the LSP refresh interval and the maximum time that an LSP can remain in the device database without a refresh.

- You can set the throttling timers for LSP generation, shortest path first computation, and partial route computation.
- You can configure the device to generate a log message when an IS-IS adjacency changes state (Up or Down).
- If a link in the network has a maximum transmission unit (MTU) size of less than 1500 bytes, you can lower the LSP MTU so that routing still occurs.
- You can use the **partition avoidance** command to prevent an area from becoming partitioned when full connectivity is lost among a Level 1-2 border device, adjacent Level 1 devices, and end hosts.

IS-IS Interface Parameters

You can optionally configure certain interface-specific IS-IS parameters independently from other attached devices. However, if you change default value, such as multipliers and time intervals, it makes sense to also change them on multiple devices and interfaces. Most of the interface parameters can be configured for level 1, level 2, or both.

The following are the interface-level parameters that you can configure:

- The default metric on the interface that is used as a value for the IS-IS metric and assigned when quality of service (QoS) routing is not performed.
- The hello interval (length of time between hello packets sent on the interface) or the default hello packet multiplier used on the interface to determine the hold time sent in IS-IS hello packets. The hold time determines how long a neighbor waits for another hello packet before declaring the neighbor down. This determines how quickly a failed link or neighbor is detected so that routes can be recalculated. Change the hello multiplier in circumstances where hello packets are lost frequently and IS-IS adjacencies are failing unnecessarily. You can raise the hello multiplier and lower the hello interval correspondingly to make the hello protocol more reliable, without increasing the time required to detect a link failure.
- Other time intervals:
  - Complete sequence number PDU (CSNP) interval—CSNPs are sent by the designated device to maintain database synchronization.
  - Retransmission interval—This is the time between retransmission of IS-IS LSPs for point-to-point links.
  - IS-IS LSP retransmission throttle interval—This is the maximum rate (number of milliseconds between packets) at which IS-IS LSPs are resent on point-to-point links. This interval is different from the retransmission interval, which is the time between successive retransmissions of the same LSP.
- Designated device-election priority, which allows you to reduce the number of adjacencies required on a multiaccess network, which in turn reduces the amount of routing protocol traffic and the size of the topology database.
- The interface circuit type, which is the type of adjacency required for neighbors on the specified interface.
- Password authentication for the interface.

How to Configure IS-IS

The following sections provide information on how to enable IS-IS on an interface, how to configure IS-IS global parameters, and how to configure IS-IS interface parameters.

Default IS-IS Configuration

Table 28: Default IS-IS Configuration

Feature	Default Setting
Ignore link-state PDU (LSP) errors	Enabled.
IS-IS type	Conventional IS-IS—The router acts as both a Level 1 (station) and a router. Multiarea IS-IS—The first instance of the IS-IS routing process is a router. Remaining instances are Level 1 routers.
Default-information originate	Disabled.
Log IS-IS adjacency state changes.	Disabled.
LSP generation throttling timers	Maximum interval between two consecutive occurrences—5000 mi Initial LSP generation delay—50 milliseconds. Hold time between the first and second LSP generation—200 millis
LSP maximum lifetime (without a refresh)	1200 seconds (20 minutes) before the LSP packet is deleted.
LSP refresh interval	Every 900 seconds (15 minutes).
Maximum LSP packet size	1497 bytes.
NSF Awareness	Enabled. Allows Layer 3 devicess to continue forwarding packets from Nonstop Forwarding-capable router during hardware or software ch
Partial route computation (PRC) throttling timers	Maximum PRC wait interval—5000 milliseconds. Initial PRC calculation delay after a topology change—50 milliseco Hold time between the first and second PRC calculation—200 milli
Partition avoidance	Disabled.
Password	No area or domain password is defined, and authentication is disabl
Set-overload-bit	Disabled. When enabled, if no arguments are entered, the overload b immediately and remains set until you enter the **no set-overload-bit**

Feature	Default Setting
Shortest path first (SPF) throttling timers	Maximum interval between consecutive SFPs—5000 milliseconds. Initial SFP calculation after a topology change—200 milliseconds. Hold time between the first and second SFP calculation—50 millisecon
Summary-address	Disabled.

Enabling IS-IS Routing

To enable IS-IS, specify a name and a network entity title (NET) for each routing process. Enable IS-IS routing on the interface and specify the area for each instance of the routing process.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**router isis** [*area tag*] **Example:** `Device(config)# router isis tag1`	Enables IS-IS routing for the specified routing process and enters IS-IS routing configuration mode. (Optional) Use the *area tag* argument to identify the area to which the IS-IS router is assigned. Enter a value if you are configuring multiple IS-IS areas. The first IS-IS instance that is configured is Level 1-2 by default. Later instances are automatically configured as Level 1. You can change the level of routing by using the **is-type** command in global configuration mode.
Step 3	**net** *network-entity-title* **Example:** `Device(config-router)# net 47.0004.004d.0001.0001.0c11.1111.00`	Configures the NETs for the routing process. While configuring multiarea IS-IS, specify a NET for each routing process. Specify a name for a NET and for an address.
Step 4	**is-type** {**level-1** | **level-1-2** | **level-2-only**} **Example:** `Device(config-router)# is-type level-2-only`	(Optional) Configures the router to act as a Level 1 (station) router, a Level 2 (area) router for multiarea routing, or both (the default): • **level 1**—Acts as a station router only. • **level 1-2**—Acts as both a station router and an area router. • **level 2**—Acts as an area router only.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config-router)# end`	Returns to global configuration mode.
Step 6	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Specifies an interface to route IS-IS, and enters interface configuration mode. If the interface is not already configured as a Layer 3 interface, enter the **no switchport** command to configure the interface into Layer 3 mode.
Step 7	**ip router isis** [*area tag*] **Example:** `Device(config-if)# ip router isis tag1`	Configures an IS-IS routing process on the interface and attaches an area designator to the routing process.
Step 8	**ip address** *ip-address-mask* **Example:** `Device(config-if)# ip address 10.0.0.5 255.255.255.0`	Defines the IP address for the interface. An IP address is required for all the interfaces in an area, that is enabled for IS-IS, if any one interface is configured for IS-IS routing.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 10	**show isis** [*area tag*] **database detail** **Example:** `Device# show isis database detail`	Verifies your entries.

Configuring IS-IS Global Parameters

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router isis** **Example:** `Device(config)# router isis`	Specifies the IS-IS routing protocol and enters router configuration mode.

	Command or Action	Purpose
Step 3	**default-information originate** [**route-map** *map-name*] **Example:** `Device(config-router)# default-information originate route-map map1`	(Optional) Forces a default route into the IS-IS routing domain. If you enter **route-map** *map-name*, the routing process generates the default route if the route map is satisfied.
Step 4	**ignore-lsp-errors** **Example:** `Device(config-router)# ignore-lsp-errors`	(Optional) Configures the router to ignore LSPs with internal checksum errors, instead of purging the LSPs. This command is enabled by default (corrupted LSPs are dropped). To purge the corrupted LSPs, enter the **no ignore-lsp-errors** router configuration command.
Step 5	**area-password** *password* **Example:** `Device(config-router)# area-password 1password`	(Optional Configures the area authentication password that is inserted in Level 1 (station router level) LSPs.
Step 6	**domain-password** *password* **Example:** `Device(config-router)# domain-password 2password`	(Optional) Configures the routing domain authentication password that is inserted in Level 2 (area router level) LSPs.
Step 7	**summary-address** *address mask* [**level-1** | **level-1-2** | **level-2**] **Example:** `Device(config-router)# summary-address 10.1.0.0 255.255.0.0 level-2`	(Optional) Creates a summary of addresses for a given level.
Step 8	**set-overload-bit** [**on-startup** {*seconds* | **wait-for-bgp**}] **Example:** `Device(config-router)# set-overload-bit on-startup wait-for-bgp`	(Optional) Sets an overload bit to allow other routers to ignore the router in their shortest path first (SPF) calculations if the router is having problems. • (Optional) **on-startup**—Sets the overload bit only on startup. If **on-startup** is not specified, the overload bit is set immediately and remains set until you enter the **no set-overload-bit** command. If **on-startup** is specified, you must either enter number of seconds or enter **wait-for-bgp**. • *seconds*—When the **on-startup** keyword is configured, it causes the overload bit to be set when the system is started and remains set for the specified number of seconds. The range is from 5 to 86400 seconds. • **wait-for-bgp**—When the **on-startup** keyword is configured, it causes the overload bit to be set when the system is started and remains set until BGP has converged. If BGP does not signal the IS-IS that it is

	Command or Action	Purpose
		converged, the IS-IS will turn off the overload bit after 10 minutes.
Step 9	**lsp-refresh-interval** *seconds* **Example:** `Device(config-router)# lsp-refresh-interval 1080`	(Optional) Sets an LSP refresh interval, in seconds. The range is from 1 to 65535 seconds. The default is to send LSP refreshes every 900 seconds (15 minutes).
Step 10	**max-lsp-lifetime** *seconds* **Example:** `Device(config-router)# max-lsp-lifetime 1000`	(Optional) Sets the maximum time that LSP packets remain in the router database without being refreshed. The range is from 1 to 65535 seconds. The default is 1200 seconds (20 minutes). After the specified time interval, the LSP packet is deleted.
Step 11	**lsp-gen-interval** [**level-1** | **level-2**] *lsp-max-wait* [*lsp-initial-wait lsp-second-wait*] **Example:** `Device(config-router)# lsp-gen-interval level-2 2 50 100`	(Optional) Sets the IS-IS LSP generation throttling timers: • *lsp-max-wait*—Maximum interval (in milliseconds) between two consecutive occurrences of an LSP being generated. The range is from 1 to 120; the default is 5000. • *lsp-initial-wait*—Initial LSP generation delay (in milliseconds). The range is from 1 to 10000; the default is 50. • *lsp-second-wait*—Hold time between the first and second LSP generation (in milliseconds). The range is from 1 to 10000; the default is 200.
Step 12	**spf-interval** [**level-1** | **level-2**] *spf-max-wait* [*spf-initial-wait spf-second-wait*] **Example:** `Device(config-router)# spf-interval level-2 5 10 20`	(Optional) Sets IS-IS SPF throttling timers. • *spf-max-wait*—Maximum interval between consecutive SFPs (in milliseconds). The range is from 1 to 120; the default is 5000. • *spf-initial-wait*—Initial SFP calculation after a topology change (in milliseconds). The range is from 1 to 10000; the default is 50. • *spf-second-wait*—Hold time between the first and second SFP calculation (in milliseconds). The range is from 1 to 10000; the default is 200.
Step 13	**prc-interval** *prc-max-wait* [*prc-initial-wait prc-second-wait*] **Example:** `Device(config-router)# prc-interval 5 10 20`	(Optional) Sets IS-IS PRC throttling timers. • *prc-max-wait*—Maximum interval (in milliseconds) between two consecutive PRC calculations. The range is from 1 to 120; the default is 5000. • *prc-initial-wait*—Initial PRC calculation delay (in milliseconds) after a topology change. The range is from 1 to 10,000; the default is 50.

	Command or Action	Purpose
		• *prc-second-wait*—Hold time between the first and second PRC calculation (in milliseconds). The range is 1 to 10,000; the default is 200.
Step 14	**log-adjacency-changes** [**all**] **Example:** `Device(config-router)# log-adjacency-changes all`	(Optional) Sets the router to log IS-IS adjacency state changes. Enter **all** to include all changes generated by events that are not related to the IS-IS hellos, including End System-to-Intermediate System PDUs and link state packets (LSPs).
Step 15	**lsp-mtu** *size* **Example:** `Device(config-router)# lsp mtu 1560`	(Optional) Specifies the maximum LSP packet size, in bytes. The range is from 128 to 4352; the default is 1497 bytes. **Note** If a link in the network has a reduced MTU size, you must change the LSP MTU size on all the devices in the network.
Step 16	**partition avoidance** **Example:** `Device(config-router)# partition avoidance`	(Optional) Causes an IS-IS Level 1-2 border router to stop advertising the Level 1 area prefix into the Level 2 backbone when full connectivity is lost among the border router, all adjacent level 1 routers, and end hosts.
Step 17	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 18	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring IS-IS Interface Parameters

To configure IS-IS interface-specific parameters, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Specifies the interface to be configured and enters interface configuration mode. If the interface is not already configured as a Layer 3 interface, enter the **no switchport** command to configure the interface into Layer 3 mode.
Step 3	**isis metric** *default-metric* [**level-1** | **level-2**] **Example:** `Device(config-if)# isis metric 15`	(Optional) Configures the metric (or cost) for the specified interface. The range is from 0 to 63; the default is 10. If no level is entered, the default is applied to both Level 1 and Level 2 routers.
Step 4	**isis hello-interval** {*seconds* | **minimal**} [**level-1** | **level-2**] **Example:** `Device(config-if)# isis hello-interval minimal`	(Optional) Specifies the length of time between the hello packets sent by the device. By default, a value that is three times the hello interval *seconds* is advertised as the *holdtime* in the hello packets sent. With smaller hello intervals, topological changes are detected faster, but there is more routing traffic. • **minimal**—Causes the system to compute the hello interval based on the hello multiplier so that the resulting hold time is 1 second. • *seconds*—Range is from 1 to 65535; default is 10 seconds.
Step 5	**isis hello-multiplier** *multiplier* [**level-1** | **level-2**] **Example:** `Device(config-if)# isis hello-multiplier 5`	(Optional) Specifies the number of IS-IS hello packets a neighbor must miss before thedevice declares the adjacency as down. The range is from 3 to 1000; default is 3. **Note** Using a smaller hello multiplier causes fast convergence, but might result in routing instability.
Step 6	**isis csnp-interval** *seconds* [**level-1** | **level-2**] **Example:** `Device(config-if)# isis csnp-interval 15`	(Optional) Configures the IS-IS complete sequence number PDU (CSNP) interval for the interface. The range is from 0 to 65535; default is 10 seconds.
Step 7	**isis retransmit-interval** *seconds* **Example:** `Device(config-if)# isis retransmit-interval 7`	(Optional) Configures the number of seconds between the retransmission of IS-IS LSPs for point-to-point links. Specify an integer that is greater than the expected round-trip delay between any two routers on the network. The range is from 0 to 65535; default is 5 seconds.
Step 8	**isis retransmit-throttle-interval** *milliseconds* **Example:** `Device(config-if)# isis retransmit-throttle-interval 4000`	(Optional) Configures the IS-IS LSP retransmission throttle interval, which is the maximum rate (number of milliseconds between packets) at which IS-IS LSPs will be resent on point-to-point links. The range is from 0 to 65535; default is determined by the **isis lsp-interval** command.

	Command or Action	Purpose
Step 9	**isis priority** *value* [**level-1** | **level-2**] **Example:** `Device(config-if)# isis priority 50`	(Optional) Configures the priority for the designated router. The range is from 0 to 127; default is 64.
Step 10	**isis circuit-type** {**level-1** | **level-1-2** | **level-2-only**} **Example:** `Device(config-if)# isis circuit-type level-1-2`	(Optional) Configures the type of adjacency required for neighbors on the specified interface (specify the interface circuit type). • **level-1**—Level 1 adjacency is established if there is at least one area address that is common to both this node and its neighbors. • **level-1-2**—Level 1 and Level 2 adjacency are established if the neighbor is also configured as both Level 1 and Level 2, and there is at least one area in common. If there is no area in common, a Level 2 adjacency is established. This is the default option. • **level 2**—Level 2 adjacency is established. If the neighbor router is a Level 1 router, no adjacency is established.
Step 11	**isis password** *password* [**level-1** | **level-2**] **Example:** `Device(config-if)# isis password secret`	(Optional) Configures the authentication password for an interface. By default, authentication is disabled. Specifying Level 1 or Level 2 enables the password only for Level 1 or Level 2 routing, respectively. If you do not specify a level, the default is Level 1 and Level 2.
Step 12	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

How to Configure IS-IS Authentication

The following sections provide information on how to generate authentication keys, how to configure IS-IS authentication for an interface, and how to configure IS-IS authentication for an instance.

Configuring Authentication Keys

You can configure multiple keys with lifetimes. To send authentication packets, the key with the latest send lifetime setting is selected. If multiple keys have the same send lifetime setting, the key is randomly selected. Use the **accept-lifetime** command for examining and accepting the authentication packets that are received. The device must be aware of these lifetimes.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**key chain** *name-of-chain* **Example:** `Device(config)# key chain key10`	Identifies a key chain, and enters key chain configuration mode.
Step 3	**key** *number* **Example:** `Device(config-keychain)# key 2000`	Identifies the key number. The range is from 0 to 65535.
Step 4	**key-string** *text* **Example:** `Device(config-keychain-key)# Room 20, 10th floor`	Identifies the key string. The string can contain 1-80 uppercase and lowercase alphanumeric characters, but the first character cannot be a number.
Step 5	**accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain-key)# accept-lifetime 12:30:00 Jan 25 1009 infinite`	(Optional) Specifies the time period during which the key can be received. The *start-time* and *end-time* syntax can be either *hh:mm:ss month date year* or *hh:mm:ss date month year*. The default is forever with the default *start-time* and the earliest acceptable date is January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 6	**send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain-key)# accept-lifetime 23:30:00 Jan 25 1019 infinite`	(Optional) Specifies the time period during which the key can be sent. The *start-time* and *end-time* syntax can be either *hh:mm:ss month date year* or *hh:mm:ss date month year*. The default *start-time* is **infinite** and the earliest acceptable date is January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 7	**cryptographic-algorithm** {**hmac-sha-1** | **hmac-sha-256** | **hmac-sha-384** | **hmac-sha-512** | **md5** } **Example:** `Device(config-keychain-key)# cryptographic-algorithm hmac-sha1-256`	(Optional) Specifies the cryptographic algorithm.
Step 8	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-keychain-key)# end`	
Step 9	**show key chain** **Example:** `Device# show key chain`	Displays authentication key information.

Configuring HMAC-MD5 or Clear Text Authentication for an IS-IS Instance

To achieve a smooth transition from one authentication method to another and to allow for continuous authentication of IS-IS PDUs, perform this procedure on each device that communicates in the network.

Before you begin

You should have generated an authentication string key. The same authentication string key should be configured on all the devices in the network.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router isis** [**area** *tag*] **Example:** `Device(config)# router isis 1`	Enables IS-IS as an IP routing protocol and assigns a tag to a process, if required. Enters router configuration mode.
Step 4	**authentication send-only** [**level-1** | **level-2**] **Example:** `Device(config-router)# authentication send-only`	Specifies that authentication is performed only on the PDUs that are being sent (not received) for the specified IS-IS instance.
Step 5	**authentication mode** {**md5** | **text**}[**level-1** | **level-2**] **Example:** `Device(config-router)# authentication mode md5`	Specifies the types of authentication to be used in PDUs for the specified IS-IS instance: • **md5**—MD5 authentication. • **text**—Clear text authentication.

	Command or Action	Purpose
Step 6	**authentication key-chain** *name-of-chain* [**level-1** | **level-2**] **Example:** `Device(config-router)# authentication key-chain remote3754`	Enables authentication for the specified IS-IS instance.
Step 7	**no authentication send-only** **Example:** `Device(config-router)# no authentication send-only`	Specifies that authentication is performed only on the PDUs that are being sent and received for the specified IS-IS instance.

Configuring HMAC-MD5 or Clear Text Authentication for an IS-IS Interface

To achieve a smooth transition from one authentication method to another and to allow for continuous authentication of IS-IS PDUs, perform this procedure on each device that communicates in the network.

Before you begin

You should have generated an authentication string key. The same authentication string key should be configured on all the devices in the network.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 0`	Configures an interface.
Step 4	**isis authentication send-only** [**level-1** | **level-2**] **Example:** `Device(config-if)# isis authentication send-only`	Specifies that authentication is performed only on the PDUs being sent (not received) for the specified IS-IS interface.
Step 5	**isis authentication mode** {**md5** | **text**}[**level-1** | **level-2**] **Example:**	Specifies the types of authentication to be used in PDUs for the specified IS-IS interface:

	Command or Action	Purpose
	`Device(config-if)# isis authentication mode md5`	• **md5**—MD5 authentication. • **text**—Clear text authentication.
Step 6	**isis authentication key-chain** *name-of-chain* [**level-1** | **level-2**] **Example:** `Device(config-if)# isis authentication key-chain multistate87723`	Enables MD5 authentication for the specified IS-IS interface.
Step 7	**no isis authentication send-only** **Example:** `Device(config-if)# no isis authentication send-only`	Specifies that authentication is performed only on the PDUs that are being sent and received for the IS-IS interface.

Monitoring and Maintaining IS-IS

You can display specific IS-IS statistics, such as the contents of routing tables, caches, and databases. You can also display information about specific interfaces, filters, or neighbors.

The following table lists the privileged EXEC commands for clearing and displaying IS-IS routing.

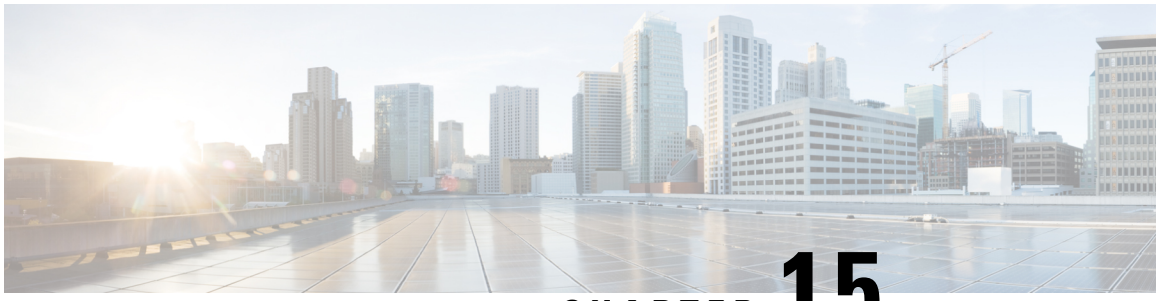
Table 29: IS-IS show Commands

Command
show ip route isis
show isis database
show isis routes
show isis spf-log
show isis topology
show route-map
trace clns *destination*

Feature Information for IS-IS

Table 30: Feature Information for IS-IS

Feature Name	Release	Feature Information
Intermediate System-to-Intermediate System (IS-IS)	Cisco IOS XE Everest 16.6.1	This feature was introduced.
	Cisco IOS XE Gibraltar 16.10.1	IS-IS now supports Secure Hash Algorithm (SHA) authentication—SHA-1, SHA-256, SHA-384, and SHA-512.



CHAPTER 15

Configuring Multi-VRF CE

Information About Multi-VRF CE

Virtual Private Networks (VPNs) provide a secure way for customers to share bandwidth over an ISP backbone network. A VPN is a collection of sites sharing a common routing table. A customer site is connected to the service-provider network by one or more interfaces, and the service provider associates each interface with a VPN routing table, called a VPN routing/forwarding (VRF) table.

The switch supports multiple VPN routing/forwarding (multi-VRF) instances in customer edge (CE) devices (multi-VRF CE) when the it is running the . Multi-VRF CE allows a service provider to support two or more VPNs with overlapping IP addresses.

Note The switch does not use Multiprotocol Label Switching (MPLS) to support VPNs.

Understanding Multi-VRF CE

Multi-VRF CE is a feature that allows a service provider to support two or more VPNs, where IP addresses can be overlapped among the VPNs. Multi-VRF CE uses input interfaces to distinguish routes for different VPNs and forms virtual packet-forwarding tables by associating one or more Layer 3 interfaces with each VRF. Interfaces in a VRF can be either physical, such as Ethernet ports, or logical, such as VLAN SVIs, but an interface cannot belong to more than one VRF at any time.

Note Multi-VRF CE interfaces must be Layer 3 interfaces.

Multi-VRF CE includes these devices:

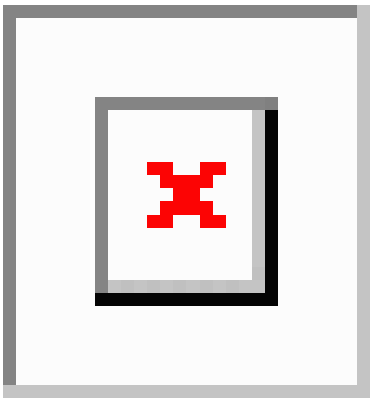
- Customer edge (CE) devices provide customers access to the service-provider network over a data link to one or more provider edge routers. The CE device advertises the site's local routes to the router and learns the remote VPN routes from it. A switch can be a CE.
- Provider edge (PE) routers exchange routing information with CE devices by using static routing or a routing protocol such as BGP, RIPv2, OSPF, or EIGRP. The PE is only required to maintain VPN routes for those VPNs to which it is directly attached, eliminating the need for the PE to maintain all of the service-provider VPN routes. Each PE router maintains a VRF for each of its directly connected sites. Multiple interfaces on a PE router can be associated with a single VRF if all of these sites participate in the same VPN. Each VPN is mapped to a specified VRF. After learning local VPN routes from CEs, a PE router exchanges VPN routing information with other PE routers by using internal BGP (IBPG).
- Provider routers or core routers are any routers in the service provider network that do not attach to CE devices.

With multi-VRF CE, multiple customers can share one CE, and only one physical link is used between the CE and the PE. The shared CE maintains separate VRF tables for each customer and switches or routes packets for each customer based on its own routing table. Multi-VRF CE extends limited PE functionality to a CE device, giving it the ability to maintain separate VRF tables to extend the privacy and security of a VPN to the branch office.

Network Topology

The figure shows a configuration using switches as multiple virtual CEs. This scenario is suited for customers who have low bandwidth requirements for their VPN service, for example, small companies. In this case, multi-VRF CE support is required in the switches. Because multi-VRF CE is a Layer 3 feature, each interface in a VRF must be a Layer 3 interface.

Figure 7: Switches Acting as Multiple Virtual CEs



When the CE switch receives a command to add a Layer 3 interface to a VRF, it sets up the appropriate mapping between the VLAN ID and the policy label (PL) in multi-VRF-CE-related data structures and adds the VLAN ID and PL to the VLAN database.

When multi-VRF CE is configured, the Layer 3 forwarding table is conceptually partitioned into two sections:

- The multi-VRF CE routing section contains the routes from different VPNs.
- The global routing section contains routes to non-VPN networks, such as the Internet.

VLAN IDs from different VRFs are mapped into different policy labels, which are used to distinguish the VRFs during processing. For each new VPN route learned, the Layer 3 setup function retrieves the policy label by using the VLAN ID of the ingress port and inserts the policy label and new route to the multi-VRF CE routing section. If the packet is received from a routed port, the port internal VLAN ID number is used; if the packet is received from an SVI, the VLAN number is used.

Packet-Forwarding Process

This is the packet-forwarding process in a multi-VRF-CE-enabled network:

- When the switch receives a packet from a VPN, the switch looks up the routing table based on the input policy label number. When a route is found, the switch forwards the packet to the PE.
- When the ingress PE receives a packet from the CE, it performs a VRF lookup. When a route is found, the router adds a corresponding MPLS label to the packet and sends it to the MPLS network.
- When an egress PE receives a packet from the network, it strips the label and uses the label to identify the correct VPN routing table. Then it performs the normal route lookup. When a route is found, it forwards the packet to the correct adjacency.
- When a CE receives a packet from an egress PE, it uses the input policy label to look up the correct VPN routing table. If a route is found, it forwards the packet within the VPN.

Network Components

To configure VRF, you create a VRF table and specify the Layer 3 interface associated with the VRF. Then configure the routing protocols in the VPN and between the CE and the PE. BGP is the preferred routing protocol used to distribute VPN routing information across the provider's backbone. The multi-VRF CE network has three major components:

- VPN route target communities—lists of all other members of a VPN community. You need to configure VPN route targets for each VPN community member.
- Multiprotocol BGP peering of VPN community PE routers—propagates VRF reachability information to all members of a VPN community. You need to configure BGP peering in all PE routers within a VPN community.
- VPN forwarding—transports all traffic between all VPN community members across a VPN service-provider network.

VRF-Aware Services

IP services can be configured on global interfaces, and these services run within the global routing instance. IP services are enhanced to run on multiple routing instances; they are VRF-aware. Any configured VRF in the system can be specified for a VRF-aware service.

VRF-Aware services are implemented in platform-independent modules. VRF means multiple routing instances in Cisco IOS. Each platform has its own limit on the number of VRFs it supports.

VRF-aware services have the following characteristics:

- The user can ping a host in a user-specified VRF.
- ARP entries are learned in separate VRFs. The user can display Address Resolution Protocol (ARP) entries for specific VRFs.

How to Configure Multi-VRF CE

Default Multi-VRF CE Configuration

Table 31: Default VRF Configuration

Feature	Default Setting
VRF	Disabled. No VRFs are defined.
Maps	No import maps, export maps, or route maps are defined.
VRF maximum routes	Fast Ethernet switches: 8000 Gigabit Ethernet switches: 12000.
Forwarding table	The default for an interface is the global routing table.

Multi-VRF CE Configuration Guidelines

- A switch with multi-VRF CE is shared by multiple customers, and each customer has its own routing table.
- Because customers use different VRF tables, the same IP addresses can be reused. Overlapped IP addresses are allowed in different VPNs.
- Multi-VRF CE lets multiple customers share the same physical link between the PE and the CE. Trunk ports with multiple VLANs separate packets among customers. Each customer has its own VLAN.
- Multi-VRF CE does not support all MPLS-VRF functionality. It does not support label exchange, LDP adjacency, or labeled packets.
- For the PE router, there is no difference between using multi-VRF CE or using multiple CEs. In Figure 41-6, multiple virtual Layer 3 interfaces are connected to the multi-VRF CE device.
- The switch supports configuring VRF by using physical ports, VLAN SVIs, or a combination of both. The SVIs can be connected through an access port or a trunk port.
- A customer can use multiple VLANs as long as they do not overlap with those of other customers. A customer's VLANs are mapped to a specific routing table ID that is used to identify the appropriate routing tables stored on the switch.
- The switch supports one global network and up to 256 VRFs.
- Most routing protocols (BGP, OSPF, RIP, and static routing) can be used between the CE and the PE. However, we recommend using external BGP (EBGP) for these reasons:
  - BGP does not require multiple algorithms to communicate with multiple CEs.
  - BGP is designed for passing routing information between systems run by different administrations.
  - BGP makes it easy to pass attributes of the routes to the CE.
- Multi-VRF CE does not affect the packet switching rate.

- VPN multicast is not supported.
- You can enable VRF on a private VLAN, and the reverse.
- You cannot enable VRF when policy-based routing (PBR) is enabled on an interface, and the reverse.
- You cannot enable VRF when Web Cache Communication Protocol (WCCP) is enabled on an interface, and the reverse.

Configuring VRFs

Perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing.
Step 3	**ip vrf** *vrf-name* **Example:** `Device(config)# ip vrf vpn1`	Names the VRF, and enter VRF configuration mode.
Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 100:2`	Creates a VRF table by specifying a route distinguisher. Enter either an AS number and an arbitrary number (xxx:y) or an IP address and arbitrary number (A.B.C.D:y)
Step 5	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf)# route-target both 100:2`	Creates a list of import, export, or import and export route target communities for the specified VRF. Enter either an AS system number and an arbitrary number (xxx:y) or an IP address and an arbitrary number (A.B.C.D:y). The *route-target-ext-community* should be the same as the *route-distinguisher* entered in Step 4.
Step 6	**import map** *route-map* **Example:** `Device(config-vrf)# import map importmap1`	(Optional) Associates a route map with the VRF.

	Command or Action	Purpose
Step 7	**interface** *interface-id* **Example:** `Device(config-vrf)# interface gigabitethernet 1/0/1`	Specifies the Layer 3 interface to be associated with the VRF, and enter interface configuration mode. The interface can be a routed port or SVI.
Step 8	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)# ip vrf forwarding vpn1`	Associates the VRF with the Layer 3 interface. **Note** When **ip vrf forwarding** is enabled in the Management Interface, the access point does not join.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 10	**show ip vrf** [**brief** | **detail** | **interfaces**] [*vrf-name*] **Example:** `Device# show ip vrf interfaces vpn1`	Verifies the configuration. Displays information about the configured VRFs.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring VRF-Aware Services

These services are VRF-Aware:

- ARP
- Ping
- Simple Network Management Protocol (SNMP)
- Unicast Reverse Path Forwarding (uRPF)
- Syslog
- Traceroute
- FTP and TFTP

Configuring VRF-Aware Services for ARP

Procedure

	Command or Action	Purpose
Step 1	**show ip arp vrf** *vrf-name* **Example:** `Device# show ip arp vrf vpn1`	Displays the ARP table in the specified VRF.

Configuring VRF-Aware Services for Ping

Procedure

	Command or Action	Purpose
Step 1	**ping vrf***vrf-name***ip-host** **Example:** `Device# ping vrf vpn1 ip-host`	Displays the ARP table in the specified VRF.

Configuring VRF-Aware Services for SNMP

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**snmp-server trap authentication vrf** **Example:** `Device(config)# snmp-server trap authentication vrf`	Enables SNMP traps for packets on a VRF.
Step 3	**snmp-server engineID remote** *host* **vrf** *vpn-instance engine-id string* **Example:** `Device(config)# snmp-server engineID remote 172.16.20.3 vrf vpn1 80000009030000B064EFE100`	Configures a name for the remote SNMP engine on a switch.
Step 4	**snmp-server host** *host* **vrf** *vpn-instance* **traps** *community* **Example:**	Specifies the recipient of an SNMP trap operation and specifies the VRF table to be used for sending SNMP traps.

	Command or Action	Purpose
	`Device(config)# snmp-server host 172.16.20.3 vrf vpn1 traps comaccess`	
Step 5	**snmp-server host** *host* **vrf** *vpn-instance* **informs** *community* **Example:** `Device(config)# snmp-server host 172.16.20.3 vrf vpn1 informs comaccess`	Specifies the recipient of an SNMP inform operation and specifies the VRF table to be used for sending SNMP informs.
Step 6	**snmp-server user** *user group* **remote** *host* **vrf** *vpn-instance security model* **Example:** `Device(config)# snmp-server user abcd remote 172.16.20.3 vrf vpn1 priv v2c 3des secure3des`	Adds a user to an SNMP group for a remote host on a VRF for SNMP access.
Step 7	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring VRF-Aware Servcies for NTP

Configuring VRF-aware services for NTP comprises configuring the NTP servers and the NTP client interfaces connected to the NTP servers.

Before you begin

Ensure connectivity between the NTP client and servers. Configure a valid IP address and subnet on the client interfaces that are connected to the NTP servers.

Configuring VRF-Aware Servcies for NTP on NTP Client

Perform the following steps on the client interface that is connected to the NTP server.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password, if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device#configure terminal`	
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Specifies the Layer 3 interface to be associated with the VRF, and enters the interface configuration mode.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)#vrf forwarding A`	Associates the VRF with the Layer 3 interface.
Step 5	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)#ip address 1.1.1.1 255.255.255.0`	Enter the IP address for the interface.
Step 6	**no shutdown** **Example:** `Device(config-if)#no shutdown`	Enables the interface.
Step 7	**exit** **Example:** `Device(config-if)exit`	Exits the interface configuration mode.
Step 8	**ntp authentication-key** *number* **md5** *md5-number* **Example:** `Device(config)#ntp authentication-key 1 md5 cisco123`	Defines the authentication keys. The device does not synchronize to a time source unless the source has one of these authentication keys and the key number is specified by the **ntp trusted-key number** command. **Note** The authentication key *number* and the MD5 *passowrd* must be the same on both the client and server.
Step 9	**ntp authenticate** **Example:** `Device(config)#ntp authenticate`	Enables the NTP authentication feature. NTP authentication is disabled by default.
Step 10	**ntp trusted-key** *key-number* **Example:** `Device(config)#ntp trusted-key 1`	Specifies one or more keys that an NTP server must provide in its NTP packets in order for the NTP client to synchronize to it. The range for trusted keys is from 1 to 65535. This command provides protection against accidentally synchronizing the NTP client to an NTP server that is not trusted.
Step 11	**ntp server vrf** *vrf-name* **Example:** `Device(config)#ntp server vrf A 1.1.1.2 key 1`	Configures NTP Server in the specified VRF.

Configuring VRF-Aware Services for NTP on the NTP Server

Perform the following steps on the NTP server.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ntp authentication-key** *number* **md5** *passowrd* **Example:** `Device(config)#ntp authentication-key 1 md5 cisco123`	Defines the authentication keys. The device does not synchronize to a time source unless the source has one of these authentication keys and the key number is specified by the **ntp trusted-key number** command. **Note** The authentication key *number* and the MD5 *passowrd* must be the same on both the client and server.
Step 4	**ntp authenticate** **Example:** `Device(config)#ntp authenticate`	Enables the NTP authentication feature. NTP authentication is disabled by default.
Step 5	**ntp trusted-key** *key-number* **Example:** `Device(config)#ntp trusted-key 1`	Specifies one or more keys that an NTP server must provide in its NTP packets in order for the NTP client to synchronize to it. The range for trusted keys is from 1 to 65535. This command provides protection against accidentally synchronizing the NTP client to an NTP server that is not trusted.
Step 6	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/3`	Specifies the Layer 3 interface to be associated with the VRF, and enters the interface configuration mode.
Step 7	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)#vrf forwarding A`	Associates the VRF with the Layer 3 interface.
Step 8	**ip address** *ip-address subnet-mask* **Example:**	Enter the IP address for the interface.

	Command or Action	Purpose
	`Device(config-if)#ip address 1.1.1.2 255.255.255.0`	
Step 9	**exit** **Example:** `Device(config-if)exit`	Exits the interface configuration mode.

Configuring VRF-Aware Servcies for uRPF

uRPF can be configured on an interface assigned to a VRF, and source lookup is done in the VRF table.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.

Step 2	**interface** *interface-id*
	Example:
	`Device(config)#`
	`interface gigabitethernet 1/0/1`
		Enters interface configuration mode, and specifies the Layer 3 interface to configure.

Step 3	**no switchport** **Example:** `Device(config-if)# no switchport`	Removes the interface from Layer 2 configuration mode if it is a physical interface.
Step 4	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)# ip vrf forwarding vpn2`	Configures VRF on the interface.
Step 5	**ip address** *ip-address* **Example:** `Device(config-if)# ip address 10.1.5.1`	Enters the IP address for the interface.
Step 6	**ip verify unicast reverse-path** **Example:** `Device(config-if)# ip verify unicast reverse-path`	Enables uRPF on the interface.
Step 7	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-if)# end`	

Configuring VRF-Aware RADIUS

To configure VRF-Aware RADIUS, you must first enable AAA on a RADIUS server. The switch supports the **ip vrf forwarding** *vrf-name* server-group configuration and the **ip radius source-interface** global configuration commands, as described in the Per VRF AAA Feature Guide.

Configuring VRF-Aware Services for Syslog

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**logging on** **Example:** `Device(config)# logging on`	Enables or temporarily disables logging of storage router event message.
Step 3	**logging host** *ip-address* **vrf** *vrf-name* **Example:** `Device(config)# logging host 10.10.1.0 vrf vpn1`	Specifies the host address of the syslog server where logging messages are to be sent.
Step 4	**logging buffered** *logging buffered size* **debugging** **Example:** `Device(config)# logging buffered critical 6000 debugging`	Logs messages to an internal buffer.
Step 5	**logging trap debugging** **Example:** `Device(config)# logging trap debugging`	Limits the logging messages sent to the syslog server.
Step 6	**logging facility** *facility* **Example:** `Device(config)# logging facility user`	Sends system logging messages to a logging facility.

	Command or Action	Purpose
Step 7	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring VRF-Aware Services for Traceroute

Procedure

	Command or Action	Purpose
Step 1	**traceroute vrf** *vrf-name ipaddress* **Example:** `Device(config)# traceroute vrf vpn2 10.10.1.1`	Specifies the name of a VPN VRF in which to find the destination address.

Configuring VRF-Aware Services for FTP and TFTP

So that FTP and TFTP are VRF-aware, you must configure some FTP/TFTP CLIs. For example, if you want to use a VRF table that is attached to an interface, say E1/0, you need to configure the ip tftp source-interface E1/0 or the ip ftp source-interface E1/0 command to inform TFTP or FTP server to use a specific routing table. In this example, the VRF table is used to look up the destination IP address. These changes are backward-compatible and do not affect existing behavior. That is, you can use the source-interface CLI to send packets out a particular interface even if no VRF is configured on that interface.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip ftp source-interface** *interface-type interface-number* **Example:** `Device(config)# ip ftp source-interface gigabitethernet 1/0/2`	Specifies the source IP address for FTP connections.
Step 3	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 4	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 5	**ip tftp source-interface** *interface-type interface-number* **Example:** `Device(config)# ip tftp source-interface gigabitethernet 1/0/2`	Specifies the source IP address for TFTP connections.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuring Multicast VRFs

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing mode.
Step 3	**ip vrf** *vrf-name* **Example:** `Device(config)# ip vrf vpn1`	Names the VRF, and enter VRF configuration mode.
Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 100:2`	Creates a VRF table by specifying a route distinguisher. Enter either an AS number and an arbitrary number (xxx:y) or an IP address and an arbitrary number (A.B.C.D:y)
Step 5	**route-target** {**export** | **import** | **both**} *route-target-ext-community*	Creates a list of import, export, or import and export route target communities for the specified VRF. Enter either an AS system number and an arbitrary number (xxx:y) or an

	Command or Action	Purpose
	Example: `Device(config-vrf)# route-target import 100:2`	IP address and an arbitrary number (A.B.C.D:y). The *route-target-ext-community* should be the same as the *route-distinguisher* entered in Step 4.
Step 6	**import map** *route-map* **Example:** `Device(config-vrf)# import map importmap1`	(Optional) Associates a route map with the VRF.
Step 7	**ip multicast-routing vrf** *vrf-name* **distributed** **Example:** `Device(config-vrf)# ip multicast-routing vrf vpn1 distributed`	(Optional) Enables global multicast routing for VRF table.
Step 8	**interface** *interface-id* **Example:** `Device(config-vrf)# interface gigabitethernet 1/0/2`	Specifies the Layer 3 interface to be associated with the VRF, and enter interface configuration mode. The interface can be a routed port or an SVI.
Step 9	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)# ip vrf forwarding vpn1`	Associates the VRF with the Layer 3 interface.
Step 10	**ip address** *ip-address* **mask** **Example:** `Device(config-if)# ip address 10.1.5.1 255.255.255.0`	Configures IP address for the Layer 3 interface.
Step 11	**ip pim sparse-dense mode** **Example:** `Device(config-if)# ip pim sparse-dense mode`	Enables PIM on the VRF-associated Layer 3 interface.
Step 12	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 13	**show ip vrf** [**brief** | **detail** | **interfaces**] [*vrf-name*] **Example:** `Device# show ip vrf detail vpn1`	Verifies the configuration. Displays information about the configured VRFs.

	Command or Action	Purpose
Step 14	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring a VPN Routing Session

Routing within the VPN can be configured with any supported routing protocol (RIP, OSPF, EIGRP, or BGP) or with static routing. The configuration shown here is for OSPF, but the process is the same for other protocols.

Note To configure an EIGRP routing process to run within a VRF instance, you must configure an autonomous-system number by entering the **autonomous-system** *autonomous-system-number* address-family configuration mode command.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router ospf** *process-id* **vrf** *vrf-name* **Example:** `Device(config)# router ospf 1 vrf vpn1`	Enables OSPF routing, specifies a VPN forwarding table, and enter router configuration mode.
Step 3	**log-adjacency-changes** **Example:** `Device(config-router)# log-adjacency-changes`	(Optional) Logs changes in the adjacency state. This is the default state.
Step 4	**redistribute bgp** *autonomous-system-number* **subnets** **Example:** `Device(config-router)# redistribute bgp 10 subnets`	Sets the switch to redistribute information from the BGP network to the OSPF network.
Step 5	**network** *network-number* **area** *area-id* **Example:** `Device(config-router)# network 1 area 2`	Defines a network address and mask on which OSPF runs and the area ID for that network address.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 7	**show ip ospf** *process-id* **Example:** `Device# show ip ospf 1`	Verifies the configuration of the OSPF network.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP PE to CE Routing Sessions

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 2`	Configures the BGP routing process with the AS number passed to other BGP routers, and enter router configuration mode.
Step 3	**network** *network-number* **mask** *network-mask* **Example:** `Device(config-router)# network 5 mask 255.255.255.0`	Specifies a network and mask to announce using BGP.
Step 4	**redistribute ospf** *process-id* **match internal** **Example:** `Device(config-router)# redistribute ospf 1 match internal`	Sets the switch to redistribute OSPF internal routes.
Step 5	**network** *network-number* **area** *area-id* **Example:**	Defines a network address and mask on which OSPF runs and the area ID for that network address.

	Command or Action	Purpose
	`Device(config-router)# network 5 area 2`	
Step 6	**address-family ipv4 vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv4 vrf vpn1`	Defines BGP parameters for PE to CE routing sessions, and enter VRF address-family mode.
Step 7	**neighbor** *address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 10.1.1.2 remote-as 2`	Defines a BGP session between PE and CE routers.
Step 8	**neighbor** *address* **activate** **Example:** `Device(config-router)# neighbor 10.2.1.1 activate`	Activates the advertisement of the IPv4 address family.
Step 9	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 10	**show ip bgp** [**ipv4**] [**neighbors**] **Example:** `Device# show ip bgp ipv4 neighbors`	Verifies BGP configuration.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring Multi-VRF CE

Table 32: Commands for Displaying Multi-VRF CE Information

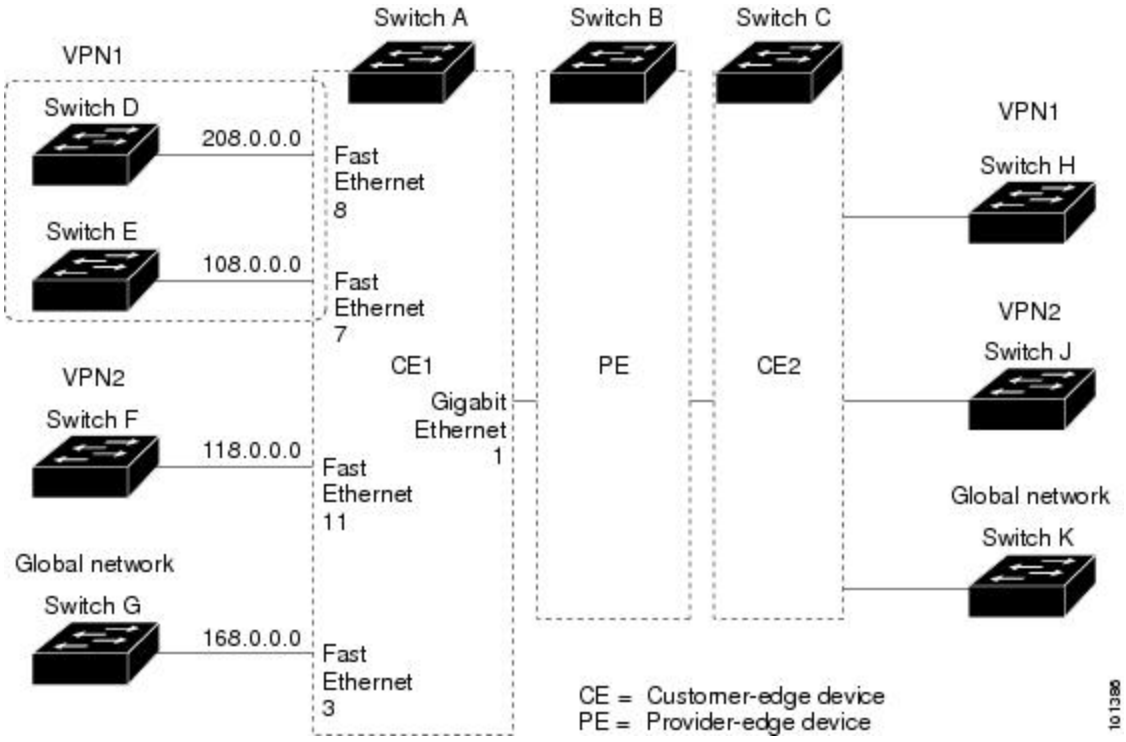
show ip protocols vrf *vrf-name*	Displays routing protocol information assoc a VRF.
show ip route vrf *vrf-name* [**connected**] [*protocol* [*as-number*]] [**list**] [**mobile**] [**odr**] [**profile**] [**static**] [**summary**] [**supernets-only**]	Displays IP routing table information assoc a VRF.
show ip vrf [**brief** | **detail** | **interfaces**] [*vrf-name*]	Displays information about the defined VRF

Configuration Examples for Multi-VRF CE

Multi-VRF CE Configuration Example

OSPF is the protocol used in VPN1, VPN2, and the global network. BGP is used in the CE to PE connections. The examples following the illustration show how to configure a switch as CE Switch A, and the VRF configuration for customer switches D and F. Commands for configuring CE Switch C and the other customer switches are not included but would be similar.

Figure 8: Multi-VRF CE Configuration Example



On Switch A, enable routing and configure VRF.

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# ip routing
Device(config)# ip vrf v11
Device(config-vrf)# rd 800:1
Device(config-vrf)# route-target export 800:1
Device(config-vrf)# route-target import 800:1
Device(config-vrf)# exit
Device(config)# ip vrf v12
Device(config-vrf)# rd 800:2
Device(config-vrf)# route-target export 800:2
Device(config-vrf)# route-target import 800:2
Device(config-vrf)# exit
```

Configure the loopback and physical interfaces on Switch A. Gigabit Ethernet port 1 is a trunk connection to the PE. Gigabit Ethernet ports 8 and 11 connect to VPNs:

```
Device(config)# interface loopback1
Device(config-if)# ip vrf forwarding v11
Device(config-if)# ip address 8.8.1.8 255.255.255.0
Device(config-if)# exit

Device(config)# interface loopback2
Device(config-if)# ip vrf forwarding v12
Device(config-if)# ip address 8.8.2.8 255.255.255.0
Device(config-if)# exit

Device(config)# interface gigabitethernet1/0/5
Device(config-if)# switchport trunk encapsulation dot1q
Device(config-if)# switchport mode trunk
Device(config-if)# no ip address
Device(config-if)# exit
Device(config)# interface gigabitethernet1/0/8
Device(config-if)# switchport access vlan 208
Device(config-if)# no ip address
Device(config-if)# exit
Device(config)# interface gigabitethernet1/0/11
Device(config-if)# switchport trunk encapsulation dot1q
Device(config-if)# switchport mode trunk
Device(config-if)# no ip address
Device(config-if)# exit
```

Configure the VLANs used on Switch A. VLAN 10 is used by VRF 11 between the CE and the PE. VLAN 20 is used by VRF 12 between the CE and the PE. VLANs 118 and 208 are used for the VPNs that include Switch F and Switch D, respectively:

```
Device(config)# interface vlan10
Device(config-if)# ip vrf forwarding v11
Device(config-if)# ip address 38.0.0.8 255.255.255.0
Device(config-if)# exit
Device(config)# interface vlan20
Device(config-if)# ip vrf forwarding v12
Device(config-if)# ip address 83.0.0.8 255.255.255.0
Device(config-if)# exit
Device(config)# interface vlan118
Device(config-if)# ip vrf forwarding v12
Device(config-if)# ip address 118.0.0.8 255.255.255.0
Device(config-if)# exit
Device(config)# interface vlan208
Device(config-if)# ip vrf forwarding v11
Device(config-if)# ip address 208.0.0.8 255.255.255.0
Device(config-if)# exit
```

Configure OSPF routing in VPN1 and VPN2.

```
Device(config)# router ospf 1 vrf vl1
Device(config-router)# redistribute bgp 800 subnets
Device(config-router)# network 208.0.0.0 0.0.0.255 area 0
Device(config-router)# exit
Device(config)# router ospf 2 vrf vl2
Device(config-router)# redistribute bgp 800 subnets
Device(config-router)# network 118.0.0.0 0.0.0.255 area 0
Device(config-router)# exit
```

Configure BGP for CE to PE routing.

```
Device(config)# router bgp 800
Device(config-router)# address-family ipv4 vrf vl2
Device(config-router-af)# redistribute ospf 2 match internal
Device(config-router-af)# neighbor 83.0.0.3 remote-as 100
Device(config-router-af)# neighbor 83.0.0.3 activate
Device(config-router-af)# network 8.8.2.0 mask 255.255.255.0
Device(config-router-af)# exit
Device(config-router)# address-family ipv4 vrf vl1
Device(config-router-af)# redistribute ospf 1 match internal
Device(config-router-af)# neighbor 38.0.0.3 remote-as 100
Device(config-router-af)# neighbor 38.0.0.3 activate
Device(config-router-af)# network 8.8.1.0 mask 255.255.255.0
Device(config-router-af)# end
```

Switch D belongs to VPN 1. Configure the connection to Switch A by using these commands.

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# ip routing
Device(config)# interface gigabitethernet1/0/2
Device(config-if)# no switchport
Device(config-if)# ip address 208.0.0.20 255.255.255.0
Device(config-if)# exit

Device(config)# router ospf 101
Device(config-router)# network 208.0.0.0 0.0.0.255 area 0
Device(config-router)# end
```

Switch F belongs to VPN 2. Configure the connection to Switch A by using these commands.

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# ip routing
Device(config)# interface gigabitethernet1/0/1
Device(config-if)# switchport trunk encapsulation dot1q
Device(config-if)# switchport mode trunk
Device(config-if)# no ip address
Device(config-if)# exit

Device(config)# interface vlan118
Device(config-if)# ip address 118.0.0.11 255.255.255.0
Device(config-if)# exit

Device(config)# router ospf 101
Device(config-router)# network 118.0.0.0 0.0.0.255 area 0
Device(config-router)# end
```

When used on switch B (the PE router), these commands configure only the connections to the CE device, Switch A.

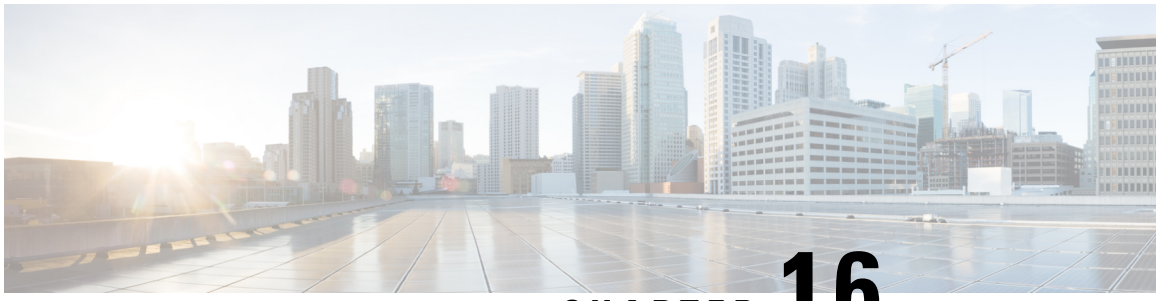
```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip vrf v1
Router(config-vrf)# rd 100:1
Router(config-vrf)# route-target export 100:1
Router(config-vrf)# route-target import 100:1
Router(config-vrf)# exit

Router(config)# ip vrf v2
Router(config-vrf)# rd 100:2
Router(config-vrf)# route-target export 100:2
Router(config-vrf)# route-target import 100:2
```

```
Router(config-vrf)# exit
Router(config)# ip cef
Router(config)# interface Loopback1
Router(config-if)# ip vrf forwarding v1
Router(config-if)# ip address 3.3.1.3 255.255.255.0
Router(config-if)# exit

Router(config)# interface Loopback2
Router(config-if)# ip vrf forwarding v2
Router(config-if)# ip address 3.3.2.3 255.255.255.0
Router(config-if)# exit

Router(config)# interface gigabitethernet1/1/0.10
Router(config-if)# encapsulation dot1q 10
Router(config-if)# ip vrf forwarding v1
Router(config-if)# ip address 38.0.0.3 255.255.255.0
Router(config-if)# exit

Router(config)# interface gigabitethernet1/1/0.20
Router(config-if)# encapsulation dot1q 20
Router(config-if)# ip vrf forwarding v2
Router(config-if)# ip address 83.0.0.3 255.255.255.0
Router(config-if)# exit

Router(config)# router bgp 100
Router(config-router)# address-family ipv4 vrf v2
Router(config-router-af)# neighbor 83.0.0.8 remote-as 800
Router(config-router-af)# neighbor 83.0.0.8 activate
Router(config-router-af)# network 3.3.2.0 mask 255.255.255.0
Router(config-router-af)# exit
Router(config-router)# address-family ipv4 vrf vl
Router(config-router-af)# neighbor 38.0.0.8 remote-as 800
Router(config-router-af)# neighbor 38.0.0.8 activate
Router(config-router-af)# network 3.3.1.0 mask 255.255.255.0
Router(config-router-af)# end
```

Feature Information for Multi-VRF CE

Table 33: Feature Information for Multi-VRF CE

Feature Name	Release	Feature Information
Multi-VRF CE	Cisco IOS XE Everest 16.6.1	This feature was introduced



CHAPTER 16

Configuring Unicast Reverse Path Forwarding

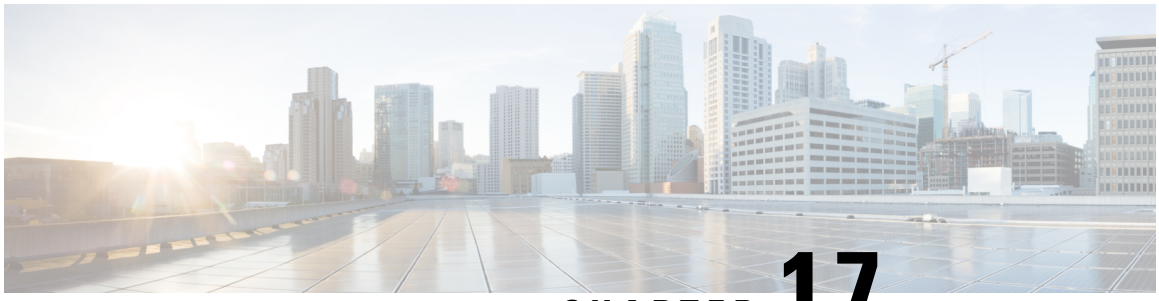
Configuring Unicast Reverse Path Forwarding

The unicast reverse path forwarding (unicast RPF) feature helps to mitigate problems that are caused by the introduction of malformed or forged (spoofed) IP source addresses into a network by discarding IP packets that lack a verifiable IP source address. For example, a number of common types of denial-of-service (DoS) attacks, including Smurf and Tribal Flood Network (TFN), can take advantage of forged or rapidly changing source IP addresses to allow attackers to thwart efforts to locate or filter the attacks. For Internet service providers (ISPs) that provide public access, Unicast RPF deflects such attacks by forwarding only packets that have source addresses that are valid and consistent with the IP routing table. This action protects the network of the ISP, its customer, and the rest of the Internet.

Note

- Unicast RPF is supported in .

Configuring Unicast Reverse Path Forwarding



CHAPTER 17

Protocol-Independent Features

Protocol-Independent Features

This section describes IP routing protocol-independent features that are available on switches running the feature set .

Distributed Cisco Express Forwarding

Information About Cisco Express Forwarding

Cisco Express Forwarding (CEF) is a Layer 3 IP switching technology used to optimize network performance. CEF implements an advanced IP look-up and forwarding algorithm to deliver maximum Layer 3 switching performance. CEF is less CPU-intensive than fast switching route caching, allowing more CPU processing power to be dedicated to packet forwarding. In a switch stack, the hardware uses distributed CEF (dCEF) in the stack. In dynamic networks, fast switching cache entries are frequently invalidated because of routing changes, which can cause traffic to be process switched using the routing table, instead of fast switched using the route cache. CEF and dCEF use the Forwarding Information Base (FIB) lookup table to perform destination-based switching of IP packets.

The two main components in CEF and dCEF are the distributed FIB and the distributed adjacency tables.

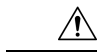
- The FIB is similar to a routing table or information base and maintains a mirror image of the forwarding information in the IP routing table. When routing or topology changes occur in the network, the IP routing table is updated, and those changes are reflected in the FIB. The FIB maintains next-hop address information based on the information in the IP routing table. Because the FIB contains all known routes that exist in the routing table, CEF eliminates route cache maintenance, is more efficient for switching traffic, and is not affected by traffic patterns.
- Nodes in the network are said to be adjacent if they can reach each other with a single hop across a link layer. CEF uses adjacency tables to prepend Layer 2 addressing information. The adjacency table maintains Layer 2 next-hop addresses for all FIB entries.

Because the switch or switch stack uses Application Specific Integrated Circuits (ASICs) to achieve Gigabit-speed line rate IP traffic, CEF or dCEF forwarding applies only to the software-forwarding path, that is, traffic that is forwarded by the CPU.

How to Configure Cisco Express Forwarding

CEF or distributed CEF is enabled globally by default. If for some reason it is disabled, you can re-enable it by using the **ip cef** or **ip cef distributed** global configuration command.

The default configuration is CEF or dCEF enabled on all Layer 3 interfaces. Entering the **no ip route-cache cef** interface configuration command disables CEF for traffic that is being forwarded by software. This command does not affect the hardware forwarding path. Disabling CEF and using the **debug ip packet detail** privileged EXEC command can be useful to debug software-forwarded traffic. To enable CEF on an interface for the software-forwarding path, use the **ip route-cache cef** interface configuration command.



Caution Although the **no ip route-cache cef** interface configuration command to disable CEF on an interface is visible in the CLI, we strongly recommend that you do not disable CEF or dCEF on interfaces except for debugging purposes.

To enable CEF or dCEF globally and on an interface for software-forwarded traffic if it has been disabled:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 2	**ip cef** **Example:** `Device(config)# ip cef`	Enables CEF operation on a non-stacking switch. Go to Step 4.
Step 3	**ip cef distributed** **Example:** `Device(config)# ip cef distributed`	Enables CEF operation on a active switch.
Step 4	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 5	**ip route-cache cef** **Example:** `Device(config-if)# ip route-cache cef`	Enables CEF on the interface for software-forwarded traffic.
Step 6	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-if)# end`	
Step 7	**show ip cef** **Example:** `Device# show ip cef`	Displays the CEF status on all interfaces.
Step 8	**show cef linecard** [**detail**] **Example:** `Device# show cef linecard detail`	(Optional) Displays CEF-related interface information on a non-stacking switch.
Step 9	**show cef linecard** [*slot-number*] [**detail**] **Example:** `Device# show cef linecard 5 detail`	(Optional) Displays CEF-related interface information on a switch by stack member for all switches in the stack or for the specified switch. (Optional) For *slot-number*, enter the stack member switch number.
Step 10	**show cef interface** [*interface-id*] **Example:** `Device# show cef interface gigabitethernet 1/0/1`	Displays detailed CEF information for all interfaces or the specified interface.
Step 11	**show adjacency** **Example:** `Device# show adjacency`	Displays CEF adjacency table information.
Step 12	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Load-Balancing Scheme for CEF Traffic

Restrictions for Configuring a Load-Balancing Scheme for CEF Traffic

- You must globally configure load balancing on device or device stack members in the same way.
- Per-packet load balancing for CEF traffic is not supported.

CEF Load-Balancing Overview

CEF load balancing allows you to optimize resources by distributing traffic over multiple paths. CEF load balancing works based on a combination of source and destination packet information.

You can configure load balancing on a per-destination. Because load-balancing decisions are made on the outbound interface, load balancing must be configured on the outbound interface.

Per-Destination Load Balancing for CEF Traffic

Per-destination load balancing allows the device to use multiple paths to achieve load sharing across multiple source-destination host pairs. Packets for a given source-destination host pair are guaranteed to take the same path, even if multiple paths are available. Traffic streams destined for different pairs tend to take different paths.

Per-destination load balancing is enabled by default when you enable CEF. To use per-destination load balancing, you do not perform any additional tasks once CEF is enabled. Per-destination is the load-balancing method of choice for most situations.

Because per-destination load balancing depends on the statistical distribution of traffic, load sharing becomes more effective as the number of source-destination host pairs increases.

You can use per-destination load balancing to ensure that packets for a given host pair arrive in order. All packets intended for a certain host pair are routed over the same link (or links).

Load-Balancing Algorithms for CEF Traffic

The following load-balancing algorithms are provided for use with CEF traffic. Select a load-balancing algorithm with the **ip cef load-sharing algorithm** command.

- Original algorithm—The original load-balancing algorithm produces distortions in load sharing across multiple devices because the same algorithm was used on every device. Depending on your network environment, you should select the algorithm.
- Universal algorithm—The universal load-balancing algorithm allows each device on the network to make a different load sharing decision for each source-destination address pair, which resolves load-sharing imbalances. The device is set to perform universal load sharing by default.

How to Configure a Load-Balancing for CEF Traffic

The following sections provide information on configuring load-balancing for CEF traffic.

Enabling or Disabling CEF Per-Destination Load Balancing

To enable or disable CEF per-destination load balancing, perform the following procedure:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-id*
4. [**no**] **ip load-sharing per-destination**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enters global configuration mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config-if)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	[**no**] **ip load-sharing per-destination** **Example:** `Device(config-if)# ip load-sharing per-destination`	Enables per-destination load balancing for CEF on the interface. The **no ip load-sharing per-destination** command disables per-destination load balancing for CEF on the interface.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Selecting a Tunnel Load-Balancing Algorithm for CEF Traffic

Select the tunnel algorithm when your network environment contains only a few source and destination pairs. The device is set to perform universal load sharing by default.

To select a tunnel load-balancing algorithm for CEF traffic, perform the following procedure:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip cef load-sharing algorithm** {**original** | **universal** [*id*] }
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enters global configuration mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip cef load-sharing algorithm** {**original** | **universal** [*id*] } **Example:** `Device(config)# ip cef load-sharing algorithm universal`	Selects a CEF load-balancing algorithm. • The **original** keyword sets the load-balancing algorithm to the original algorithm, based on a source IP and destination IP hash. • The **universal** keyword sets the load-balancing algorithm to one that uses a source IP, destination IP, Layer 3 Protocol, Layer 4 source port, Layer 4 destination port and IPv6 flow label (for IPv6 traffic). • The *id* argument is a fixed identifier.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuration Examples for CEF Traffic Load-Balancing

The following sections provide configuration examples for CEF traffic load-balancing.

Example: Enabling or Disabling CEF Per-Destination Load Balancing

Per-destination load balancing is enabled by default when you enable CEF. The following example shows how to disable per-destination load balancing:

```
Device> enable
Device# configure terminal
Device(config)# interface Ethernet1/0/1
Device(config-if)# no ip load-sharing per-destination
Device(config-if)# end
```

Number of Equal-Cost Routing Paths

Information About Equal-Cost Routing Paths

When a router has two or more routes to the same network with the same metrics, these routes can be thought of as having an equal cost. The term parallel path is another way to see occurrences of equal-cost routes in a routing table. If a router has two or more equal-cost paths to a network, it can use them concurrently. Parallel paths provide redundancy in case of a circuit failure and also enable a router to load balance packets over the available paths for more efficient use of available bandwidth. Equal-cost routes are supported across switches in a stack.

Even though the router automatically learns about and configures equal-cost routes, you can control the maximum number of parallel paths supported by an IP routing protocol in its routing table. Although the switch software allows a maximum of 32 equal-cost routes, the switch hardware will never use more than 16 paths per route.

How to Configure Equal-Cost Routing Paths

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router** {**rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router eigrp`	Enters router configuration mode.
Step 3	**maximum-paths** *maximum* **Example:** `Device(config-router)# maximum-paths 2`	Sets the maximum number of parallel paths for the protocol routing table. The range is from 1 to 16; the default is 4 for most IP routing protocols, but only 1 for BGP.
Step 4	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 5	**show ip protocols** **Example:** `Device# show ip protocols`	Verifies the setting in the *Maximum path* field.
Step 6	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Static Unicast Routes

Information About Static Unicast Routes

Static unicast routes are user-defined routes that cause packets moving between a source and a destination to take a specified path. Static routes can be important if the router cannot build a route to a particular destination and are useful for specifying a gateway of last resort to which all unroutable packets are sent.

The switch retains static routes until you remove them. However, you can override static routes with dynamic routing information by assigning administrative distance values. Each dynamic routing protocol has a default administrative distance, as listed in Table 41-16. If you want a static route to be overridden by information from a dynamic routing protocol, set the administrative distance of the static route higher than that of the dynamic protocol.

Table 34: Dynamic Routing Protocol Default Administrative Distances

Route Source	Default Distance
Connected interface	0
Static route	1
Enhanced IRGP summary route	5
Internal Enhanced IGRP	90
IGRP	100
OSPF	110
Unknown	225

Static routes that point to an interface are advertised through RIP, IGRP, and other dynamic routing protocols, whether or not static **redistribute** router configuration commands were specified for those routing protocols. These static routes are advertised because static routes that point to an interface are considered in the routing table to be connected and hence lose their static nature. However, if you define a static route to an interface that is not one of the networks defined in a network command, no dynamic routing protocols advertise the route unless a **redistribute** static command is specified for these protocols.

When an interface goes down, all static routes through that interface are removed from the IP routing table. When the software can no longer find a valid next hop for the address specified as the forwarding router's address in a static route, the static route is also removed from the IP routing table.

Configuring Static Unicast Routes

Static unicast routes are user-defined routes that cause packets moving between a source and a destination to take a specified path. Static routes can be important if the router cannot build a route to a particular destination and are useful for specifying a gateway of last resort to which all unroutable packets are sent.

Follow these steps to configure a static route:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**ip route prefix mask** {*address* | *interface*} [*distance*] **Example:** `Device(config)# ip route prefix mask gigabitethernet 1/0/4`	Establish a static route.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show ip route** **Example:** `Device# show ip route`	Displays the current state of the routing table to verify the configuration.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

Use the **no ip route** *prefix mask* {*address*| *interface*} global configuration command to remove a static route. The device retains static routes until you remove them.

Default Routes and Networks

Information About Default Routes and Networks

A router might not be able to learn the routes to all other networks. To provide complete routing capability, you can use some routers as smart routers and give the remaining routers default routes to the smart router. (Smart routers have routing table information for the entire internetwork.) These default routes can be dynamically learned or can be configured in the individual routers. Most dynamic interior routing protocols include a mechanism for causing a smart router to generate dynamic default information that is then forwarded to other routers.

If a router has a directly connected interface to the specified default network, the dynamic routing protocols running on that device generate a default route. In RIP, it advertises the pseudonetwork 0.0.0.0.

A router that is generating the default for a network also might need a default of its own. One way a router can generate its own default is to specify a static route to the network 0.0.0.0 through the appropriate device.

When default information is passed through a dynamic routing protocol, no further configuration is required. The system periodically scans its routing table to choose the optimal default network as its default route. In IGRP networks, there might be several candidate networks for the system default. Cisco routers use administrative distance and metric information to set the default route or the gateway of last resort.

If dynamic default information is not being passed to the system, candidates for the default route are specified with the **ip default-network** global configuration command. If this network appears in the routing table from any source, it is flagged as a possible choice for the default route. If the router has no interface on the default network, but does have a path to it, the network is considered as a possible candidate, and the gateway to the best default path becomes the gateway of last resort.

How to Configure Default Routes and Networks

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ip default-network** *network number* **Example:** `Device(config)# ip default-network 1`	Specifies a default network.
Step 3	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 4	**show ip route** **Example:**	Displays the selected default route in the gateway of last resort display.

	Command or Action	Purpose
	`Device# show ip route`	
Step 5	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Route Maps to Redistribute Routing Information

Information About Route Maps

The switch can run multiple routing protocols simultaneously, and it can redistribute information from one routing protocol to another. Redistributing information from one routing protocol to another applies to all supported IP-based routing protocols.

You can also conditionally control the redistribution of routes between routing domains by defining enhanced packet filters or route maps between the two domains. The **match** and **set** route-map configuration commands define the condition portion of a route map. The **match** command specifies that a criterion must be matched. The **set** command specifies an action to be taken if the routing update meets the conditions defined by the match command. Although redistribution is a protocol-independent feature, some of the **match** and **set** route-map configuration commands are specific to a particular protocol.

One or more **match** commands and one or more **set** commands follow a **route-map** command. If there are no **match** commands, everything matches. If there are no **set** commands, nothing is done, other than the match. Therefore, you need at least one **match** or **set** command.

Note A route map with no **set** route-map configuration commands is sent to the CPU, which causes high CPU utilization.

You can also identify route-map statements as **permit** or **deny**. If the statement is marked as a deny, the packets meeting the match criteria are sent back through the normal forwarding channels (destination-based routing). If the statement is marked as permit, set clauses are applied to packets meeting the match criteria. Packets that do not meet the match criteria are forwarded through the normal routing channel.

How to Configure a Route Map

Although each of Steps 3 through 14 in the following section is optional, you must enter at least one **match** route-map configuration command and one **set** route-map configuration command.

Note The keywords are the same as defined in the procedure to control the route distribution.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**route-map***map-tag* [**permit** | **deny**] [*sequence number*] **Example:** `Device(config)# route-map rip-to-ospf permit 4`	Defines any route maps used to control redistribution and enter route-map configuration mode. *map-tag*—A meaningful name for the route map. The **redistribute** router configuration command uses this name to reference this route map. Multiple route maps might share the same map tag name. (Optional) If **permit** is specified and the match criteria are met for this route map, the route is redistributed as controlled by the set actions. If **deny** is specified, the route is not redistributed. *sequence number* (Optional)— Number that indicates the position a new route map is to have in the list of route maps already configured with the same name.
Step 3	**match as-path** *path-list-number* **Example:** `Device(config-route-map)#match as-path 10`	Matches a BGP AS path access list.
Step 4	**match community-list** *community-list-number* [**exact**] **Example:** `Device(config-route-map)# match community-list 150`	Matches a BGP community list.
Step 5	**match ip address** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:** `Device(config-route-map)# match ip address 5 80`	Matches a standard access list by specifying the name or number. It can be an integer from 1 to 199.
Step 6	**match metric** metric-value **Example:** `Device(config-route-map)# match metric 2000`	Matches the specified route metric. The *metric-value* can be an EIGRP metric with a specified value from 0 to 4294967295.
Step 7	**match ip next-hop** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:**	Matches a next-hop router address passed by one of the access lists specified (numbered from 1 to 199).

	Command or Action	Purpose
	`Device(config-route-map)# match ip next-hop 8 45`	
Step 8	**match tag** *tag value* [*...tag-value*] **Example:** `Device(config-route-map)# match tag 3500`	Matches the specified tag value in a list of one or more route tag values. Each can be an integer from 0 to 4294967295.
Step 9	**match interface***type number* [*...type-number*] **Example:** `Device(config-route-map)# match interface gigabitethernet 1/0/1`	Matches the specified next hop route out one of the specified interfaces.
Step 10	**match ip route-source** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:** `Device(config-route-map)# match ip route-source 10 30`	Matches the address specified by the specified advertised access lists.
Step 11	**match route-type** {**local** | **internal** | **external** [**type-1** | **type-2**]} **Example:** `Device(config-route-map)# match route-type local`	Matches the specified **route-type**: • **local**—Locally generated BGP routes. • **internal**—OSPF intra-area and interarea routes or EIGRP internal routes. • **external**—OSPF external routes (Type 1 or Type 2) or EIGRP external routes.
Step 12	**set dampening** *halflife reuse suppress max-suppress-time* **Example:** `Device(config-route-map)# set dampening 30 1500 10000 120`	Sets BGP route dampening factors.
Step 13	**set local-preference** *value* **Example:** `Device(config-route-map)# set local-preference 100`	Assigns a value to a local BGP path.
Step 14	**set origin** {**igp** | **egp** *as* | **incomplete**} **Example:** `Device(config-route-map)#set origin igp`	Sets the BGP origin code.

	Command or Action	Purpose
Step 15	**set as-path** {**tag** | **prepend** *as-path-string*} **Example:** `Device(config-route-map)# set as-path tag`	Modifies the BGP autonomous system path.
Step 16	**set level** {**level-1** | **level-2** | **level-1-2** | **stub-area** | **backbone**} **Example:** `Device(config-route-map)# set level level-1-2`	Sets the level for routes that are advertised into the specified area of the routing domain. The **stub-area** and **backbone** are OSPF NSSA and backbone areas.
Step 17	**set metric** *metric value* **Example:** `Device(config-route-map)# set metric 100`	Sets the metric value to give the redistributed routes (for EIGRP only). The *metric value* is an integer from -294967295 to 294967295.
Step 18	**set metric***bandwidth delay reliability loading mtu* **Example:** `Device(config-route-map)# set metric 10000 10 255 1 1500`	Sets the metric value to give the redistributed routes (for EIGRP only): • *bandwidth*—Metric value or IGRP bandwidth of the route in kilobits per second in the range 0 to 4294967295 • *delay*—Route delay in tens of microseconds in the range 0 to 4294967295. • *reliability*—Likelihood of successful packet transmission expressed as a number between 0 and 255, where 255 means 100 percent reliability and 0 means no reliability. • *loading*—Effective bandwidth of the route expressed as a number from 0 to 255 (255 is 100 percent loading). • *mtu*—Minimum maximum transmission unit (MTU) size of the route in bytes in the range 0 to 4294967295.
Step 19	**set metric-type** {**type-1** | **type-2**} **Example:** `Device(config-route-map)# set metric-type type-2`	Sets the OSPF external metric type for redistributed routes.
Step 20	**set metric-type internal** **Example:** `Device(config-route-map)# set metric-type internal`	Sets the multi-exit discriminator (MED) value on prefixes advertised to external BGP neighbor to match the IGP metric of the next hop.

	Command or Action	Purpose
Step 21	**set weight** *number* **Example:** `Device(config-route-map)# set weight 100`	Sets the BGP weight for the routing table. The value can be from 1 to 65535.
Step 22	**end** **Example:** `Device(config-route-map)# end`	Returns to privileged EXEC mode.
Step 23	**show route-map** **Example:** `Device# show route-map`	Displays all route maps configured or only the one specified to verify configuration.
Step 24	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

How to Control Route Distribution

Although each of Steps 3 through 14 in the following section is optional, you must enter at least one **match** route-map configuration command and one **set** route-map configuration command.

Note The keywords are the same as defined in the procedure to configure the route map for redistritbution.

The metrics of one routing protocol do not necessarily translate into the metrics of another. For example, the RIP metric is a hop count, and the IGRP metric is a combination of five qualities. In these situations, an artificial metric is assigned to the redistributed route. Uncontrolled exchanging of routing information between different routing protocols can create routing loops and seriously degrade network operation.

If you have not defined a default redistribution metric that replaces metric conversion, some automatic metric translations occur between routing protocols:

- RIP can automatically redistribute static routes. It assigns static routes a metric of 1 (directly connected).
- Any protocol can redistribute other routing protocols if a default mode is in effect.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router** { **rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router eigrp 10`	Enters router configuration mode.
Step 3	**redistribute** *protocol* [*process-id*] {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**match internal** | **external** *type-value*] [**tag** *tag-value*] [**route-map** *map-tag*] [**weight** *weight*] [**subnets**] **Example:** `Device(config-router)# redistribute eigrp 1`	Redistributes routes from one routing protocol to another routing protocol. If no route-maps are specified, all routes are redistributed. If the keyword **route-map** is specified with no *map-tag*, no routes are distributed.
Step 4	**default-metric** *number* **Example:** `Device(config-router)# default-metric 1024`	Cause the current routing protocol to use the same metric value for all redistributed routes (RIP and OSPF).
Step 5	**default-metric** bandwidth delay reliability loading mtu **Example:** `Device(config-router)# default-metric 1000 100 250 100 1500`	Cause the EIGRP routing protocol to use the same metric value for all non-EIGRP redistributed routes.
Step 6	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 7	**show route-map** **Example:** `Device# show route-map`	Displays all route maps configured or only the one specified to verify configuration.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Policy-Based Routing

Restrictions for Configuring PBR

- Policy-based routing (PBR) is not supported to forward traffic into GRE tunnel. This applies to PBR applied on any interface and forwarding traffic into GRE tunnel (by means of PBR next-hop or default next-hop or set interface).

- PBR is not supported on GRE tunnel itself (applied under the GRE tunnel itself).

Information About Policy-Based Routing

You can use policy-based routing (PBR) to configure a defined policy for traffic flows. By using PBR, you can have more control over routing by reducing the reliance on routes derived from routing protocols. PBR can specify and implement routing policies that allow or deny paths based on:

- Identity of a particular end system
- Application
- Protocol

You can use PBR to provide equal-access and source-sensitive routing, routing based on interactive versus batch traffic, or routing based on dedicated links. For example, you could transfer stock records to a corporate office on a high-bandwidth, high-cost link for a short time while transmitting routine application data such as e-mail over a low-bandwidth, low-cost link.

With PBR, you classify traffic using access control lists (ACLs) and then make traffic go through a different path. PBR is applied to incoming packets. All packets received on an interface with PBR enabled are passed through route maps. Based on the criteria defined in the route maps, packets are forwarded (routed) to the appropriate next hop.

- Route map statement marked as permit is processed as follows:
  - A match command can match on length or multiple ACLs. A route map statement can contain multiple match commands. Logical or algorithm function is performed across all the match commands to reach a permit or deny decision.

    For example:

    match length A B

    match ip address acl1 acl2

    match ip address acl3

    A packet is permitted if it is permitted by match length A B or acl1 or acl2 or acl3

  - If the decision reached is permit, then the action specified by the set command is applied on the packet .
  - If the decision reached is deny, then the PBR action (specified in the set command) is not applied. Instead the processing logic moves forward to look at the next route-map statement in the sequence (the statement with the next higher sequence number). If no next statement exists, PBR processing terminates, and the packet is routed using the default IP routing table.
- For PBR, route-map statements marked as deny are not supported.

You can use standard IP ACLs to specify match criteria for a source address or extended IP ACLs to specify match criteria based on an application, a protocol type, or an end station. The process proceeds through the route map until a match is found. If no match is found, normal destination-based routing occurs. There is an implicit deny at the end of the list of match statements.

If match clauses are satisfied, you can use a set clause to specify the IP addresses identifying the next hop router in the path.

How to Configure PBR

- Multicast traffic is not policy-routed. PBR applies only to unicast traffic.
- You can enable PBR on a routed port or an SVI.
- The switch supports PBR based on match length.
- You can apply a policy route map to an EtherChannel port channel in Layer 3 mode, but you cannot apply a policy route map to a physical interface that is a member of the EtherChannel. If you try to do so, the command is rejected. When a policy route map is applied to a physical interface, that interface cannot become a member of an EtherChannel.
- You can define a mazimum of 128 IP policy route maps on the switch or switch stack.
- You can define a maximum of 512 access control entries(ACEs) for PBR on the switch or switch stack.
- When configuring match criteria in a route map, follow these guidelines:
  - Do not match ACLs that permit packets destined for a local address.
- Web Cache Communication Protocol (WCCP) and PBR are mutually exclusive on a switch interface. You cannot enable WCCP when PBR is enabled on an interface. The reverse is also true, you cannot enable PBR when WCCP is enabled on an interface.
- The number of hardware entries used by PBR depends on the route map itself, the ACLs used, and the order of the ACLs and route-map entries.
- PBR based on TOS, DSCP and IP Precedence are not supported.
- Set interface, set default next-hop and set default interface are not supported.
- **ip next-hop recursive** and **ip next-hop verify availability** features are not available and the next-hop should be directly connected.
- Policy-maps with no set actions are supported. Matching packets are routed normally.
- Policy-maps with no match clauses are supported. Set actions are applied to all packets.

By default, PBR is disabled on the switch. To enable PBR, you must create a route map that specifies the match criteria and the resulting action. Then, you must enable PBR for that route map on an interface. All packets arriving on the specified interface matching the match clauses are subject to PBR.

Packets that are generated by the switch (CPU), or local packets, are not normally policy-routed. When you globally enable local PBR on the switch, all unicast packets that originate on the switch are subject to local PBR. The protocols that are supported for local PBR are NTP, DNS, MSDP, SYSLOG and TFTP. Local PBR is disabled by default.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**route-map** *map-tag* [**permit**] [*sequence number*] **Example:** `Device(config)# route-map pbr-map permit`	Defines route maps that are used to control where packets are output, and enters route-map configuration mode. • *map-tag* — A meaningful name for the route map. The **ip policy route-map** interface configuration command uses this name to reference the route map. Multiple route-map statements with the same map tag define a single route map. • (Optional) **permit** — If **permit** is specified and the match criteria are met for this route map, the route is policy routed as defined by the set actions. • (Optional) *sequence number* — The sequence number shows the position of the route-map statement in the given route map.
Step 4	**match ip address** {*access-list-number* | *access-list-name*} [*access-list-number* |*...access-list-name*] **Example:** `Device(config-route-map)# match ip address 110 140`	Matches the source and destination IP addresses that are permitted by one or more standard or extended access lists. ACLs can match on more than one source and destination IP address. If you do not specify a **match** command, the route map is applicable to all packets.
Step 5	**match length min max** **Example:** `Device(config-route-map)# match length 64 1500`	Matches the length of the packet.
Step 6	**set ip next-hop** *ip-address* [*...ip-address*] **Example:** `Device(config-route-map)# set ip next-hop 10.1.6.2`	Specifies the action to be taken on the packets that match the criteria. Sets next hop to which to route the packet (the next hop must be adjacent).
Step 7	**exit** **Example:** `Device(config-route-map)# exit`	Returns to global configuration mode.
Step 8	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to be configured.
Step 9	**ip policy route-map** *map-tag* **Example:** `Device(config-if)# ip policy route-map pbr-map`	Enables PBR on a Layer 3 interface, and identify the route map to use. You can configure only one route map on an interface. However, you can have multiple route map entries with different sequence numbers. These entries are

	Command or Action	Purpose
		evaluated in the order of sequence number until the first match. If there is no match, packets are routed as usual.
Step 10	**ip route-cache policy** **Example:** `Device(config-if)# ip route-cache policy`	(Optional) Enables fast-switching PBR. You must enable PBR before enabling fast-switching PBR.
Step 11	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 12	**ip local policy route-map** *map-tag* **Example:** `Device(config)# ip local policy route-map local-pbr`	(Optional) Enables local PBR to perform policy-based routing on packets originating at the switch. This applies to packets generated by the switch, and not to incoming packets.
Step 13	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 14	**show route-map** [*map-name*] **Example:** `Device# show route-map`	(Optional) Displays all the route maps configured or only the one specified to verify configuration.
Step 15	**show ip policy** **Example:** `Device# show ip policy`	(Optional) Displays policy route maps attached to the interface.
Step 16	**show ip local policy** **Example:** `Device# show ip local policy`	(Optional) Displays whether or not local policy routing is enabled and, if so, the route map being used.

Filtering Routing Information

You can filter routing protocol information by performing the tasks described in this section.

Note When routes are redistributed between OSPF processes, no OSPF metrics are preserved.

Setting Passive Interfaces

To prevent other routers on a local network from dynamically learning about routes, you can use the **passive-interface** router configuration command to keep routing update messages from being sent through a router interface. When you use this command in the OSPF protocol, the interface address you specify as

passive appears as a stub network in the OSPF domain. OSPF routing information is neither sent nor received through the specified router interface.

In networks with many interfaces, to avoid having to manually set them as passive, you can set all interfaces to be passive by default by using the **passive-interface default** router configuration command and manually setting interfaces where adjacencies are desired.

Use a network monitoring privileged EXEC command such as **show ip ospf interface** to verify the interfaces that you enabled as passive, or use the **show ip interface** privileged EXEC command to verify the interfaces that you enabled as active.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router** { **rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router ospf`	Enters router configuration mode.
Step 3	**passive-interface** *interface-id* **Example:** `Device(config-router)# passive-interface gigabitethernet 1/0/1`	Suppresses sending routing updates through the specified Layer 3 interface.
Step 4	**passive-interface default** **Example:** `Device(config-router)# passive-interface default`	(Optional) Sets all interfaces as passive by default.
Step 5	**no passive-interface** *interface type* **Example:** `Device(config-router)# no passive-interface gigabitethernet1/0/3 gigabitethernet 1/0/5`	(Optional) Activates only those interfaces that need to have adjacencies sent.
Step 6	**network** *network-address* **Example:** `Device(config-router)# network 10.1.1.1`	(Optional) Specifies the list of networks for the routing process. The *network-address* is an IP address.
Step 7	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Advertising and Processing in Routing Updates

You can use the **distribute-list** router configuration command with access control lists to suppress routes from being advertised in routing updates and to prevent other routers from learning one or more routes. When used in OSPF, this feature applies to only external routes, and you cannot specify an interface name.

You can also use a **distribute-list** router configuration command to avoid processing certain routes listed in incoming updates. (This feature does not apply to OSPF.)

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router** { **rip** | **eigrp**} **Example:** `Device(config)# router eigrp 10`	Enters router configuration mode.
Step 3	**distribute-list** {*access-list-number* | *access-list-name*} **out** [*interface-name* | *routing process* | *autonomous-system-number*] **Example:** `Device(config-router)# distribute 120 out gigabitethernet 1/0/7`	Permits or denies routes from being advertised in routing updates, depending upon the action listed in the access list.
Step 4	**distribute-list** {*access-list-number* | *access-list-name*} **in** [*type-number*] **Example:** `Device(config-router)# distribute-list 125 in`	Suppresses processing in routes listed in updates.
Step 5	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Filtering Sources of Routing Information

Because some routing information might be more accurate than others, you can use filtering to prioritize information coming from different sources. An administrative distance is a rating of the trustworthiness of a routing information source, such as a router or group of routers. In a large network, some routing protocols can be more reliable than others. By specifying administrative distance values, you enable the router to intelligently discriminate between sources of routing information. The router always picks the route whose routing protocol has the lowest administrative distance.

Because each network has its own requirements, there are no general guidelines for assigning administrative distances.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router** { **rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router eigrp 10`	Enters router configuration mode.
Step 3	**distance** *weight* {*ip-address* {*ip-address mask*}} [*ip access list*] **Example:** `Device(config-router)# distance 50 10.1.5.1`	Defines an administrative distance. *weight*—The administrative distance as an integer from 10 to 255. Used alone, *weight* specifies a default administrative distance that is used when no other specification exists for a routing information source. Routes with a distance of 255 are not installed in the routing table. (Optional) *ip access list*—An IP standard or extended access list to be applied to incoming routing updates.
Step 4	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 5	**show ip protocols** **Example:**	Displays the default administrative distance for a specified routing process.

	Command or Action	Purpose
	`Device# show ip protocols`	
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Managing Authentication Keys

Key management is a method of controlling authentication keys used by routing protocols. Not all protocols can use key management. Authentication keys are available for EIGRP and RIP Version 2.

Prerequisites

Before you manage authentication keys, you must enable authentication. See the appropriate protocol section to see how to enable authentication for that protocol. To manage authentication keys, define a key chain, identify the keys that belong to the key chain, and specify how long each key is valid. Each key has its own key identifier (specified with the **key** *number* key chain configuration command), which is stored locally. The combination of the key identifier and the interface associated with the message uniquely identifies the authentication algorithm and Message Digest 5 (MD5) authentication key in use.

How to Configure Authentication Keys

You can configure multiple keys with life times. Only one authentication packet is sent, regardless of how many valid keys exist. The software examines the key numbers in order from lowest to highest, and uses the first valid key it encounters. The lifetimes allow for overlap during key changes. Note that the router must know these lifetimes.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**key chain** *name-of-chain* **Example:** `Device(config)# key chain key10`	Identifies a key chain, and enter key chain configuration mode.
Step 3	**key** *number* **Example:** `Device(config-keychain)# key 2000`	Identifies the key number. The range is 0 to 2147483647.

	Command or Action	Purpose
Step 4	**key-string** *text* **Example:** `Device(config-keychain)# Room 20, 10th floor`	Identifies the key string. The string can contain from 1 to 80 uppercase and lowercase alphanumeric characters, but the first character cannot be a number.
Step 5	**accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain)# accept-lifetime 12:30:00 Jan 25 1009 infinite`	(Optional) Specifies the time period during which the key can be received. The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 6	**send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain)# accept-lifetime 23:30:00 Jan 25 1019 infinite`	(Optional) Specifies the time period during which the key can be sent. The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 7	**end** **Example:** `Device(config-keychain)# end`	Returns to privileged EXEC mode.
Step 8	**show key chain** **Example:** `Device# show key chain`	Displays authentication key information.
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.